B.M.S. COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

Lab Record

Software Engineering and Object-Oriented Modeling

Submitted in partial fulfillment for the 5th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

Adithya Pillai

1BM22CS013

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
Mar-June 2024

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Object-Oriented Analysis and Design(22CS6PCSEO) laboratory has been carried out by Adithya Pillai (1BM22CS013) during the 5th Semester Oct 2024- Jan2025.

Signature of the Faculty Incharge:

DR. Latha N R

Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

Table of Contents

1. Hotel Management System
2. Credit Card Processing
3. Library Management System
4. Stock Maintenance System
5. Passport Automation System

Hotel Management System

1.1 Problem Statement

Develop a software solution to facilitate a computerized hotel management system for a network of hotels, featuring both manual operations by front desk staff and an automated online platform. Each hotel will have its own database to oversee bookings, customer information, and financial transactions, while the online platform will connect to a central system for managing reservations, payments, and service requests. The system must allow guests to book accommodations, reserve event spaces, access amenities, and securely complete payments, all while ensuring smooth integration with the respective hotel systems. It should accurately manage simultaneous bookings and service requests, maintain data privacy, and provide centralized functionalities such as event space reservations and restaurant access. Hotels will oversee their own internal processes, and the cost of the shared system will be allocated based on the usage of online reservations and extra services.

1.2 Software Requirements Specification Document

1. Introduction

1.1 Purpose of the Requirements Document: To specify the requirements for a Hotel Management System.

1.2 Scope of the Product

- Users are allowed to check in to the hotel only at 12 noon and check out at 2 pm.
- Prior booking of the party hall is required.
- Users below the age of 18 are not permitted to book a room unless accompanied by an adult.

1.3 Definitions, Acronyms, and Abbreviations

- AC: Air Conditioning
- EV: Electric Vehicle
- VAC: Ventilation and Air Conditioning

- OC: Occupied

1.4 References

- Taj Hotel
- JW Marriott

1.5 Overview: This document provides detailed information about the requirements and general booking procedures.

2. General Description

2.1 Product Perspective

- Hotel Staff Perspective:
    - Details about the residing and past customers of the hotel.
    - Details of allocated and available rooms.
    - Staff roles and working hours.
    - Monitor CCTV surveillance.
    - Party hall booking details.
    - Parking details.
    - Restaurant details.
- Hotel Guest Perspective:
    - Room availability and booking details.
    - Hotel amenities.
    - Party hall details.
    - Hotel restaurant details.

2.2 Product Functions

- Valet Parking
- Room Booking
- Party Hall Booking
- Pet-Friendly Services
- Security System
- Restaurant Access

- Hotel Maintenance
- Amenities:
    - Gym
    - Swimming Pool
    - Game Room
    - Spa

2.3 User Characteristics

- Customers(under whose name booking is done) of the hotel are above the age of 18 however Restaurant Customers are of all ages.
- Users also include staff and managers of the hotel.

2.4 General Constraints

- Customers must be above the age of 18 to book hotel rooms.
- Customers can check in only after 12 noon.
- Online booking requires advance payments.
- Customers must carry ID proof.

2.5 Assumptions and Dependencies

- Customers will maintain decorum at all times.
- Customers with pets will ensure that their pets are leashed at all times.
- The restaurant is open to the public and hotel customers.
- Children will be accompanied by their Guardian at all times.

3. Specific Requirements

3.1 Functional Requirements

- Room Booking System
- Party Hall Booking
- Security System
- Hotel Restaurant Facilities

3.2 Non-Functional Requirements

- Online booking must require advance payments.
- Party hall booking must be done at least one week in advance.
- Pet services and EV charging stations.
- Confidentiality of customer information.
- Ease and performance-friendly booking of hotel rooms and other amenities.

3.3 Domain Requirements

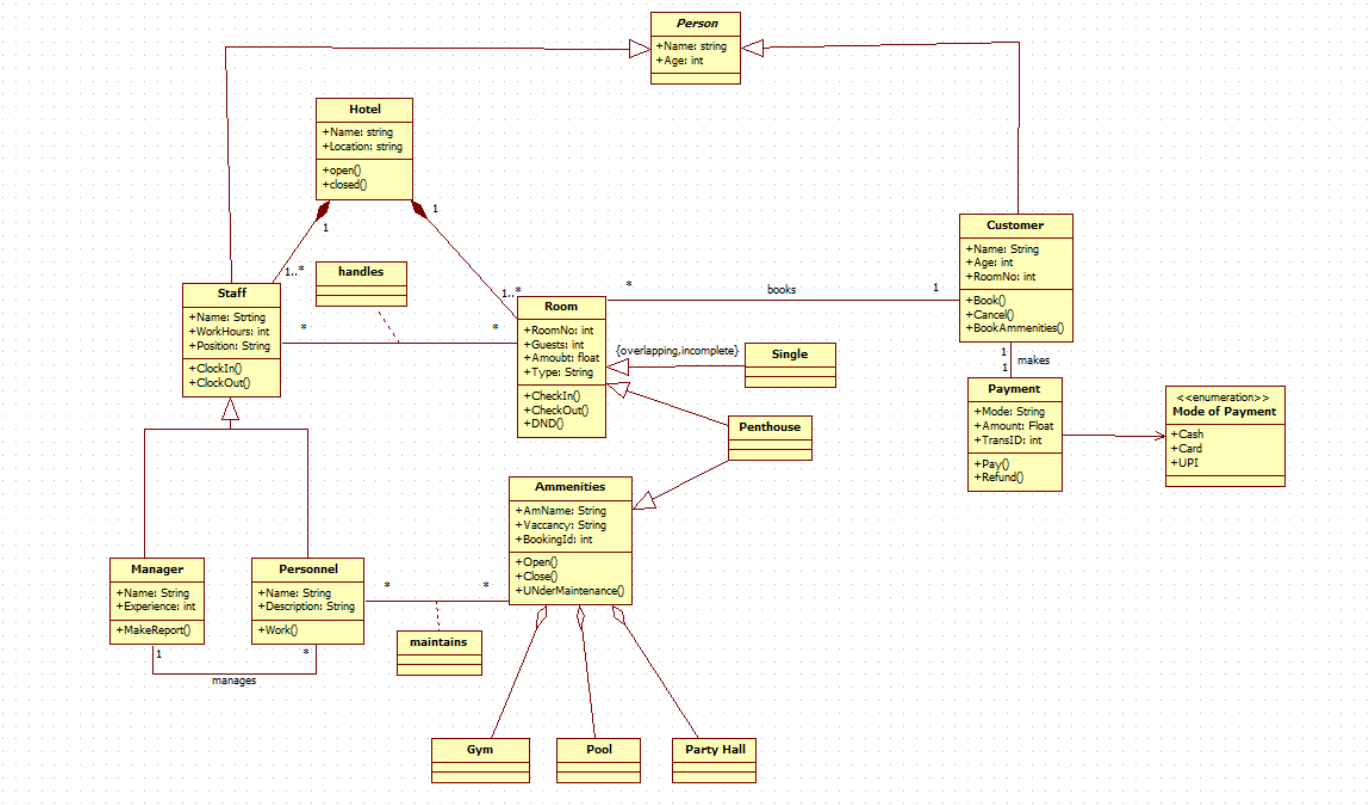
- User Interface
- Payment Gateway
- Inventory Management for the hotel
- Customer Feedback and Reviews

4. Appendices

5. Index

1.3 Class Diagram

1.3.1 Advanced Class Diagram



Description:

This diagram represents the class structure of a hotel management system.

Key Classes:

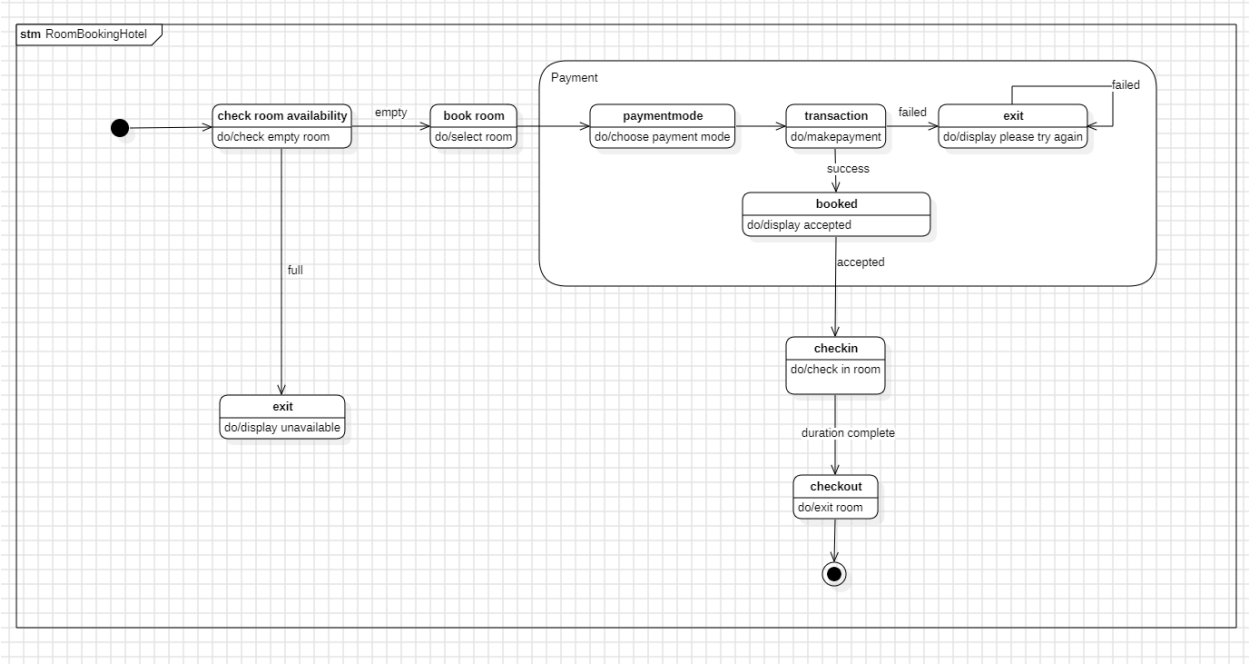
- Person: A base class with attributes Name and Age.
- Hotel: Manages rooms and amenities using methods like open() and close().
- Room: Attributes include RoomNo, Guests, Amount, Type, and methods for checking in/out.
    - Single and Penthouse: Specialized room types inheriting from Room.
- Customer: A subclass of Person with methods like Book() and Cancel().
- Payment: Handles transactions with attributes like Mode and Amount.
- Amenities: Includes facilities like Gym, Pool, and Party Hall, managed by the hotel.
- Staff: Handles hotel operations, with subcategories:
    - Manager: Handles reports and staff management.
    - Personnel: General staff with a Work() method.

Relationships:

- Customers book rooms and amenities.
- Staff maintains hotel operations.
- Payments are associated with room bookings.

1.4 State Diagram

1.4.1 Simple State Diagram



Description:

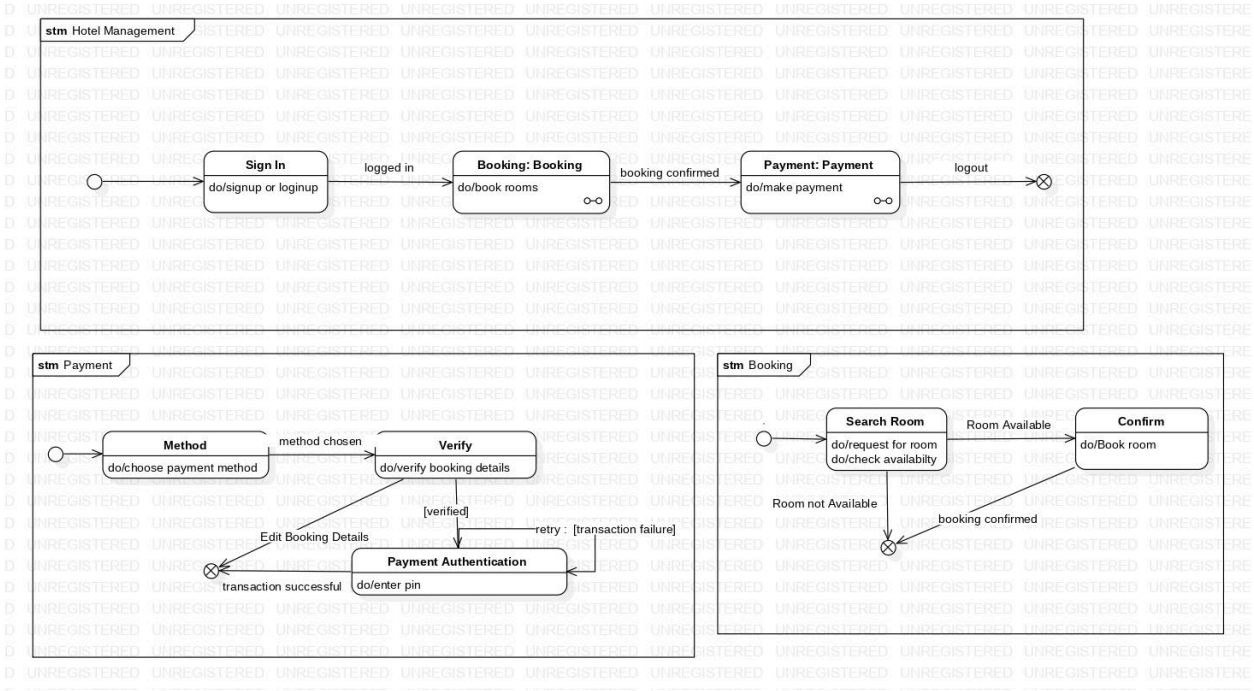
States:

- Initial: Checking room availability.
- Intermediate: Booking, payment processing, and checking in.
- Final: Room checkout and completion of stay.

Transitions:

- Payment can lead to success (booking) or failure (exit).
- After checking in, guests proceed to the checkout phase post-duration completion.

1.4.2 Advanced State Diagram



Description:

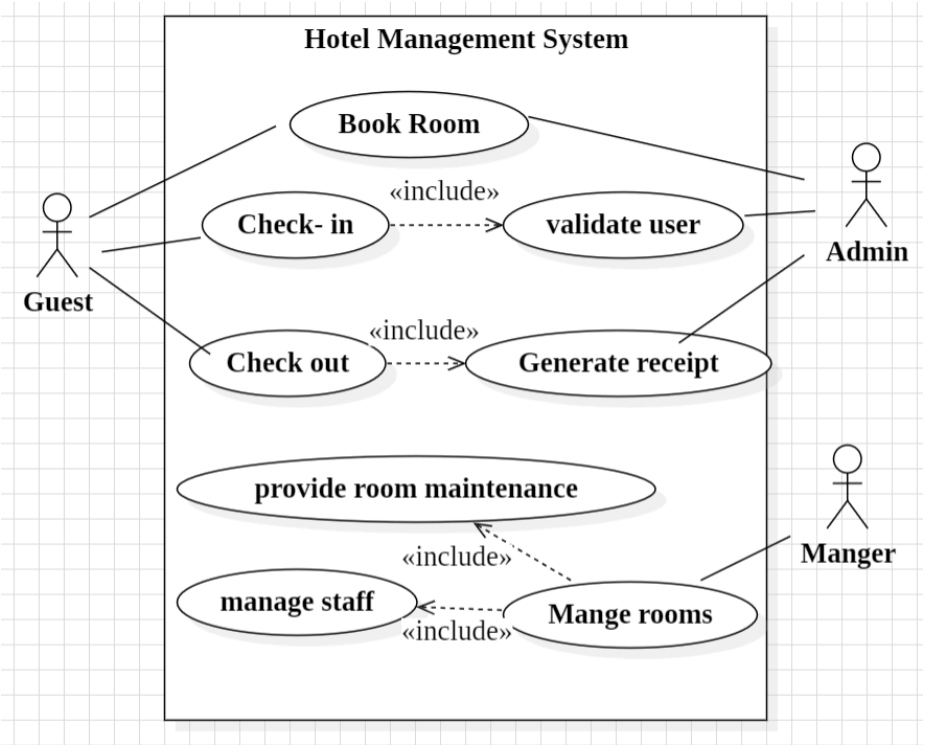
Key States and Transitions:

1. Initial State:
    - The diagram starts with an initial black circle, representing the starting point of the process.
2. States:
    - Idle State: Represents the system or object (e.g., a room) being available or awaiting action.
    - Registration: The state where the guest creates a new account.
    - Login: The state where credentials are verified.
    - Room Selection: A state where the guest chooses a room.
    - Booking: The state where room booking details are filled in and submitted.
    - Payment Processing:
        - The system transitions into processing payments after booking confirmation.
        - It involves communicating with the Payment Gateway.
    - Confirmation: The state where the system confirms the payment and booking.
    - Logout: A state for ending the session.
    - Maintenance (if applicable): For rooms or amenities undergoing maintenance.

3. Transitions:
    - Triggered by events such as login success, payment confirmation, or room availability checks.
    - Specific actions (like sending payment signals, filling booking details) move the system from one state to another.
4. Final State:
    - The diagram ends with a filled black circle, representing the termination of the process, such as after the user logs out.

1.5 Use case diagram

1.5.1 Simple Use Case Diagram

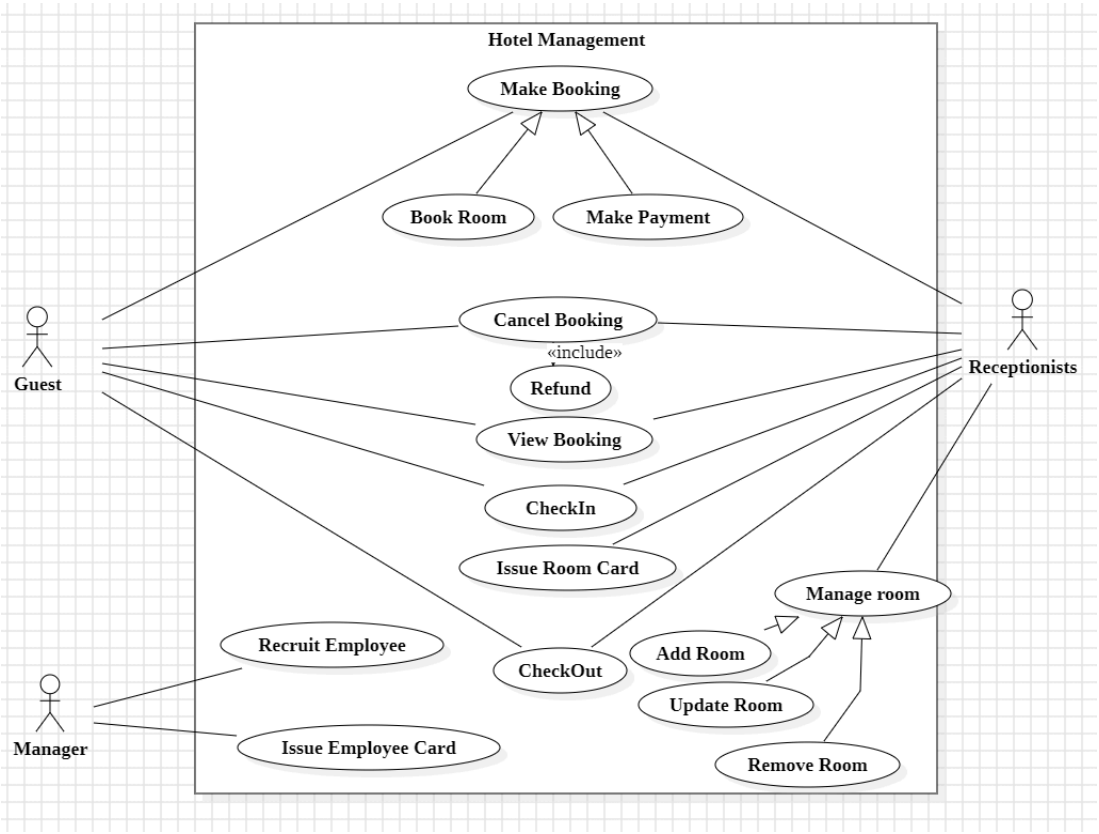


Description: This diagram outlines the interactions between different actors and the system. It identifies the primary use cases:

1. Guest: Can book a room, check-in, and check out. Booking includes validating the user and generating a receipt.
2. Admin: Validates user details.
3. Manager: Manages rooms and staff, and oversees room maintenance.

The relationships between use cases are shown using include relationships, which indicate shared functionality such as validation and receipt generation.

1.5.2 Advanced Use Case Diagram



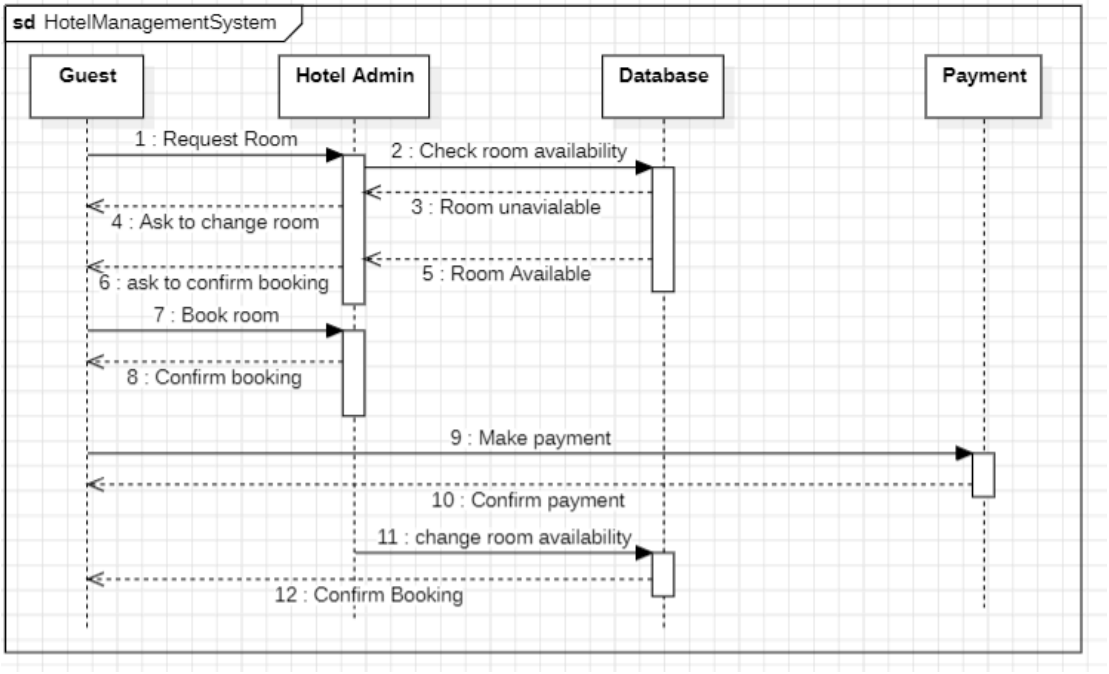
Description:

This diagram depicts the functionalities of a hotel management system and the roles interacting with it:

- Guest: Can make a booking, make payment, cancel booking (which includes refund), view booking, check-in, check-out, and issue a room card.
- Receptionists: Assist guests in booking, managing bookings, issuing room cards, and checking in/out. They also manage rooms (adding, updating, or removing).
- Manager: Handles employee recruitment and issuing employee cards.
- System Scope: Demonstrates the collaboration between different actors (guest, receptionist, manager) and their actions within the hotel management system.

1.6 Sequence Diagram

1.6.1 Simple Sequence Diagram

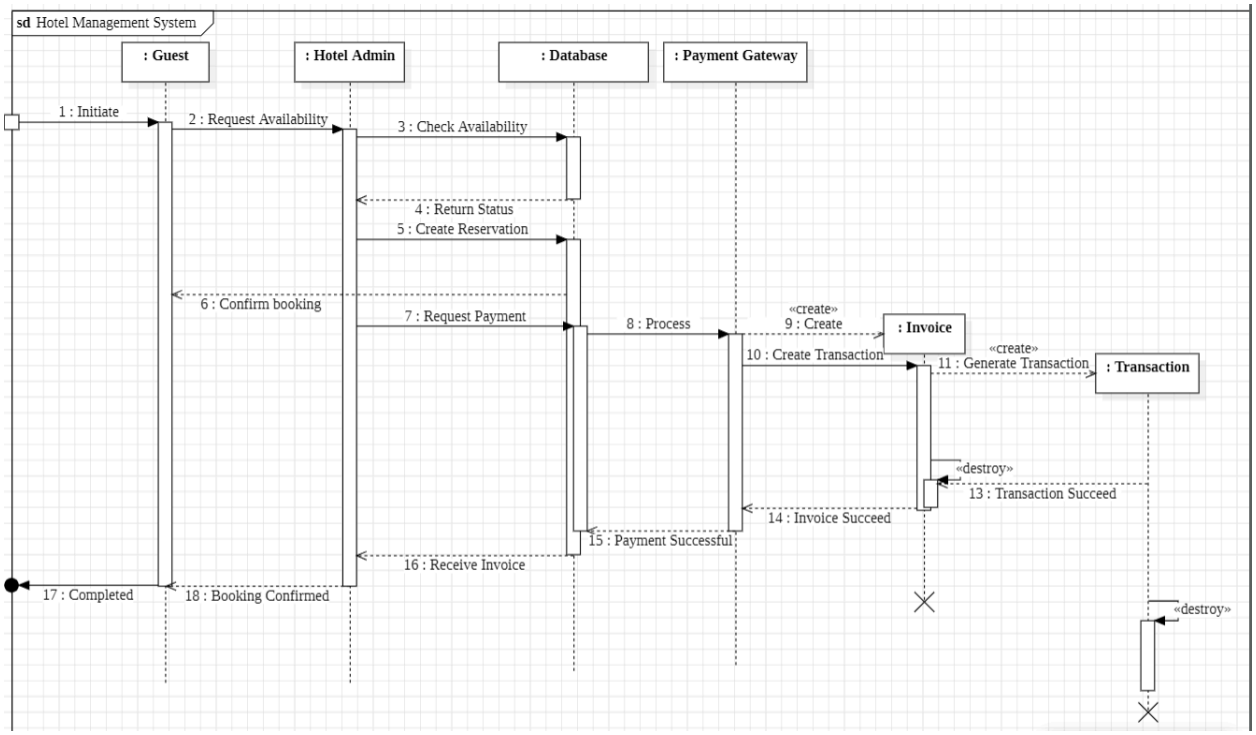


Description:

1. Guest requests a room.
2. Hotel Admin checks room availability in the Database.
    - If unavailable, the admin suggests changes.
    - If available, the admin proceeds to confirm the booking.
3. Payment is made and confirmed through the Payment system.
4. The database is updated to reflect the room's availability status.
5. The final booking is confirmed.

This captures the step-by-step interactions to fulfill a guest's booking request.

1.6.2 Advanced Sequence Diagram



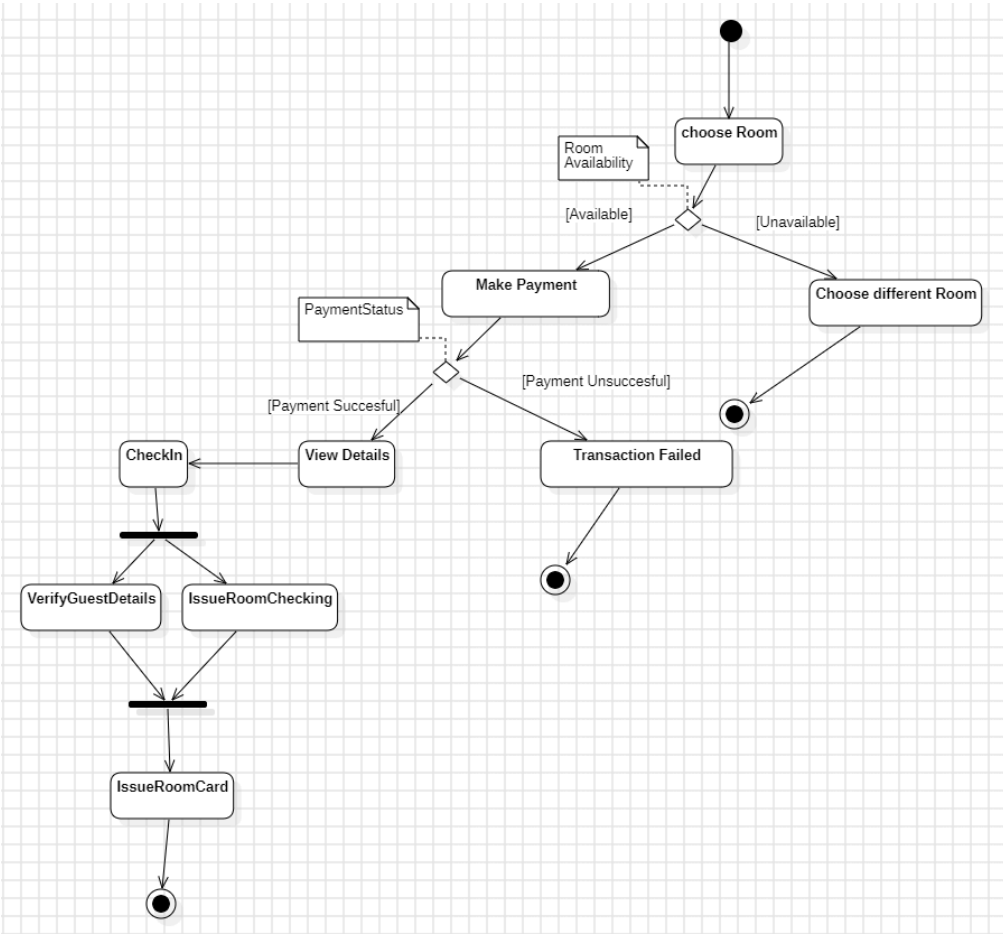
Description:

This diagram explains the flow of events when a guest interacts with the hotel management system:

1. Guest Initiates Request: The guest starts by requesting room availability.
2. Hotel Admin Checks Availability: Admin queries the database for room availability.
3. Reservation Process:
    - Availability status is returned.
    - A reservation is created in the database.
4. Payment Process:
    - Payment is requested via the payment gateway.
    - Payment gateway processes the transaction and generates an invoice.
    - Transaction success or failure is communicated back to the system.
5. Booking Confirmation: Once payment is successful, the guest is notified with a booking confirmation.

1.7 Activity Diagram

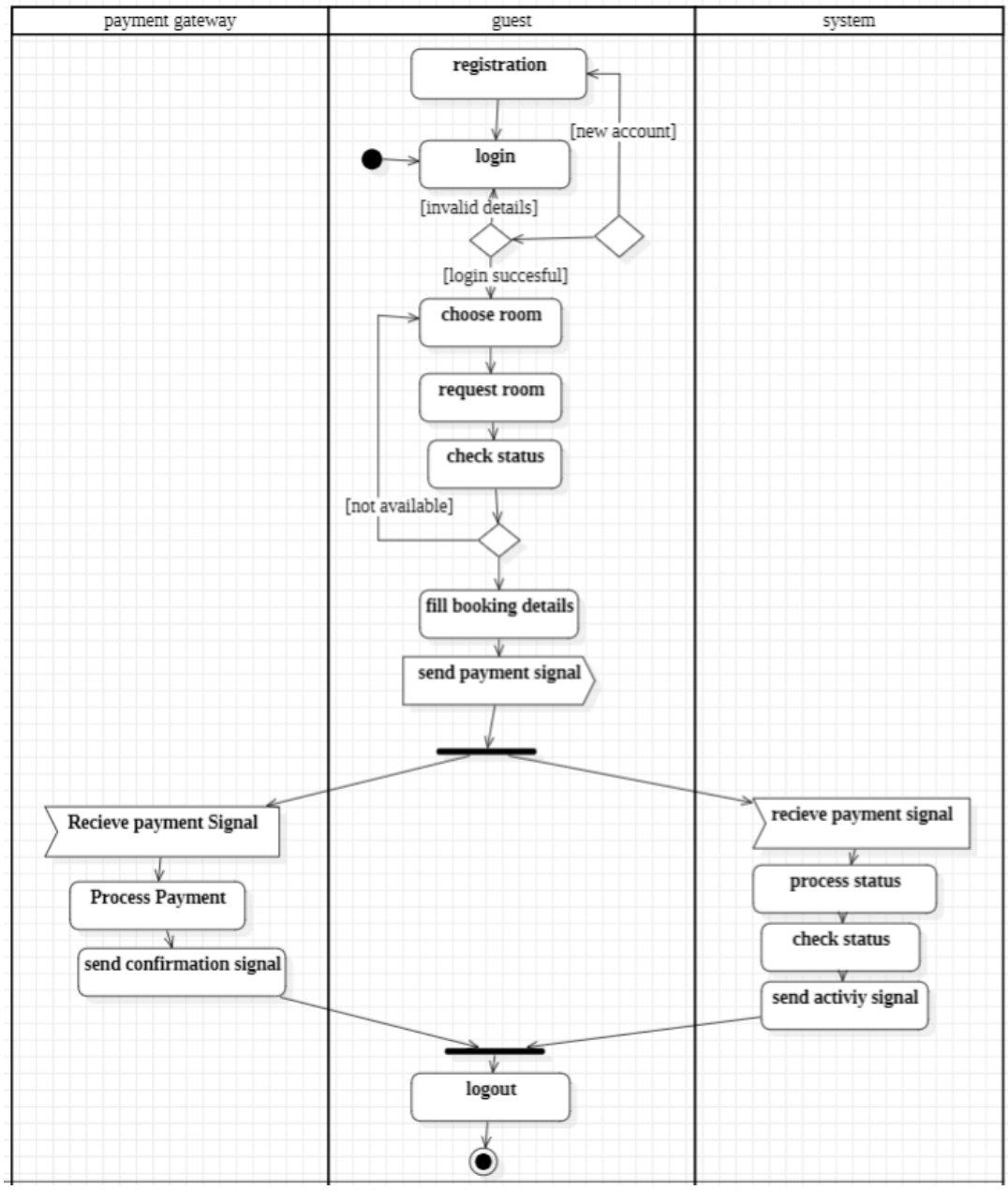
1.7.1 Simple Activity Diagram



Description: This diagram represents the workflow of the room booking and check-in process:

1. Starts with the guest choosing a room.
2. Decision points check room availability:
    - If available, the guest proceeds to make payment.
    - If unavailable, the guest can choose a different room.
3. Successful payment leads to verification and issuing of room details.
4. Unsuccessful payment results in a transaction failure.
5. The check-in process includes verifying guest details and issuing a room card, completing the activity.

1.7.2 Advanced Activity Diagram



Description:

This diagram illustrates the step-by-step process of a hotel booking system using an activity diagram. The main swimlanes are:

- Payment Gateway: Handles payment transactions.
- Guest: Represents a user interacting with the system.
- System: The hotel booking system managing the core logic.

Key Activities:

1. Registration: A guest registers for a new account.
2. Login: The user logs in, with checks for valid credentials.
3. Choose and Request Room: Once logged in, the guest chooses and requests a room. If unavailable, the process repeats.
4. Fill Booking Details: After room confirmation, the user fills in booking details.
5. Send Payment Signal: The system processes payment via a gateway.
6. Confirmation: The payment is processed and confirmed.
7. Logout: The guest logs out after completing the process.

Credit Card Processing System

2.1 Problem Statement

Design a secure system for processing credit card transactions in real-time across physical point-of-sale devices, online platforms, and mobile gateways. The system should verify card information, approve payments, and manage refunds and chargebacks while implementing fraud detection and adhering to PCI-DSS standards. Merchants will keep transaction records, while a centralized system oversees communication with banks and card networks for both authorization and settlement. The system needs to accommodate various card types, manage high transaction volumes, and offer user-friendly interfaces. Costs will be allocated based on transaction volumes and service utilization. The software is intended to provide efficient, secure, and scalable payment processing solutions for both merchants and customers.

2.2 SRS Document

1. Introduction

1.1 Purpose of the Requirements Document: To specify the software requirements for the Credit Card Processing System.

1.2 Scope of the Product: Allows customers to make payments at merchant locations using credit cards. The system handles the authorization, capture, and payment of transactions.

1.3 Definitions, Acronyms, and Abbreviations

- POS: Point of Sale
- API: Application Programming Interface
- CVV: Card Verification Value
- EMV: Europay, MasterCard, and Visa
- PCI-DSS: Payment Card Industry Data Security Standard

1.4 References: Guidelines from PayPal, Visa, and Mastercard Merchant Services.

1.5 Overview: This document includes functional and non-functional aspects, constraints, and domain-specific features. It also describes the processes of transaction initiation, validation, authorization, and settlement.

2. General Description

2.1 Product Perspective: The system interacts with external banking networks, internal merchant systems, POS devices, mobile payment gateways, and online e-commerce platforms.

- Bank Perspective: Interfaces with banks and card networks to authorize transactions, check for fraud, and ensure funds availability.
- Merchant Perspective: Provides tools for merchants to process credit card payments.
- Customer Perspective: Enables customers to securely make payments using credit cards, receive real-time authorization responses, and view their transaction status.

2.2 Product Functions

- Transaction authorization
- Transaction capture
- Transaction settlement
- Fraud detection
- Chargeback management
- Refund processing

2.3 User Characteristics

- Merchants: Business owners or staff processing payments at physical or online stores.
- Customers: Individuals making credit card payments.
- Banks: Financial institutions that authorize transactions and provide merchant services.

2.4 General Constraints

- Must comply with PCI-DSS standards for data security.
- System should support all major card types.
- Transactions above ₹15,000/- require additional authorization checks.

2.5 Assumptions and Dependencies

- Merchants will ensure customers input correct payment details.
- The system ensures constant connectivity with banks and card networks for real-time authorization.

3. Specific Requirements

3.1 Functional Requirements

- Transaction Authorization:
    - Validate card number, expiration date, CVV, and address.
    - Ensure enough funds are available for the transaction.
    - Provide real-time response to merchants and customers.
- Fraud Detection: Cross-check transactions for unusual patterns based on user history.
- Chargeback Handling: Automatically reverse payments when disputes are resolved in favor of the customer.
- Refund Management: Enable partial or full refunds for authorized transactions.

3.2 Non-Functional Requirements

- Security: Must comply with PCI-DSS standards to ensure data protection.
- Performance:
    - System should handle up to 500 transactions per second during peak times.
    - Average response time for authorization should be less than 2 seconds.
- Reliability: System uptime should be 99.9% with a failover mechanism in place.

3.3 Domain Requirements

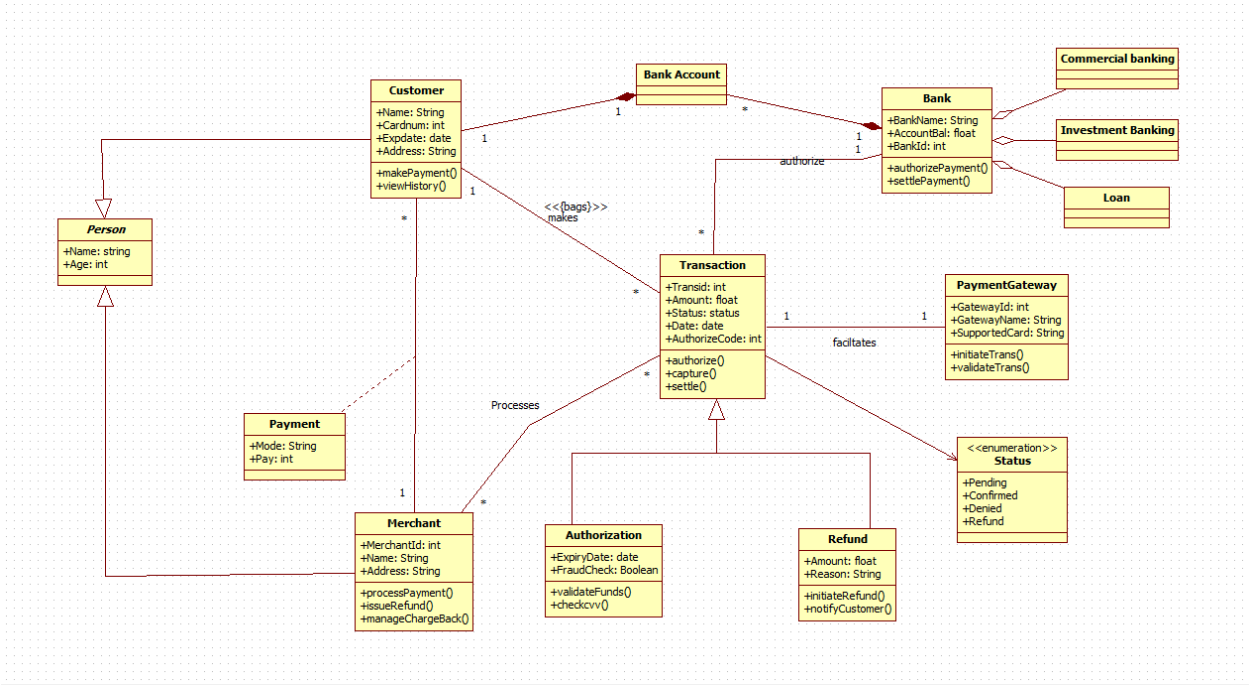
- Responsive user interface for mobile and desktop users.
- Inventory management.
- Support multiple payment gateways.

4. Appendix

5. Index

2.3 Class Diagram

2.3.1 Advanced Class Diagram



Description:

Classes

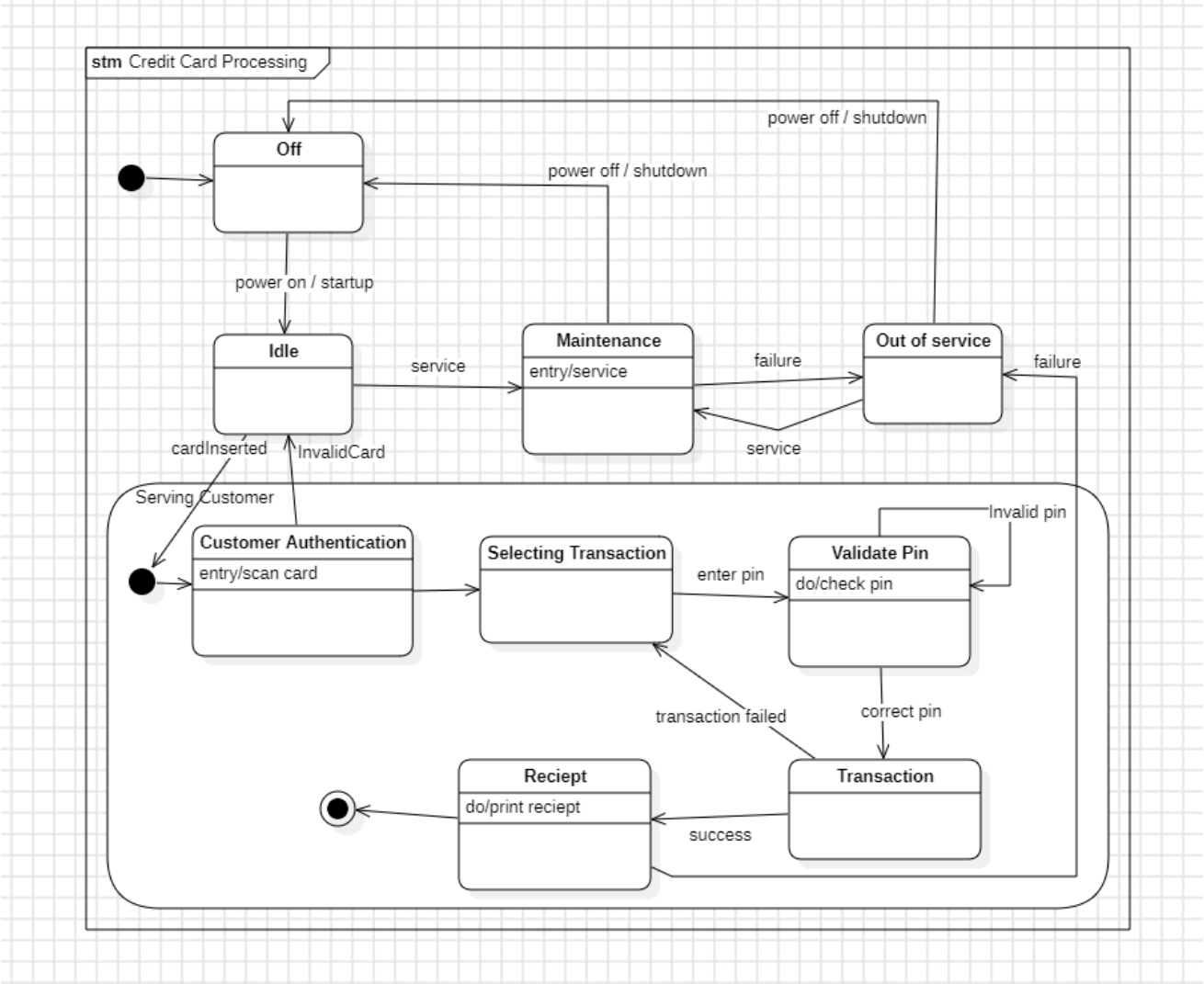
1. Person: Base class for individuals.
2. Customer:
    - Inherits from Person.
    - Represents bank customers with details like address and card information.
3. Bank: Represents banks with attributes like name and ID.
4. Bank Account:
    - Linked to Bank.
    - Represents accounts with attributes like balance and account holder details.
5. Transaction:
    - Represents financial transactions.
    - Attributes include amount, date, and status.
6. Payment: Represents payment methods used in transactions.
7. Merchant: Represents businesses accepting payments.
8. Payment Gateway: Facilitates payment processing.
9. Authorization: Represents transaction authorizations.
10. Refund: Represents refunds for transactions.

Relationships

1. Inheritance: Customer inherits from Person.
2. Aggregation: Bank Account is aggregated by Bank.
3. Association: Customer is associated with Bank Account.
4. Composition: Transaction is composed of Payment.
5. Dependency: Transaction depends on Payment Gateway.
6. Enumeration: Status defines states for transactions (e.g., pending, confirmed).

2.4 State Diagram

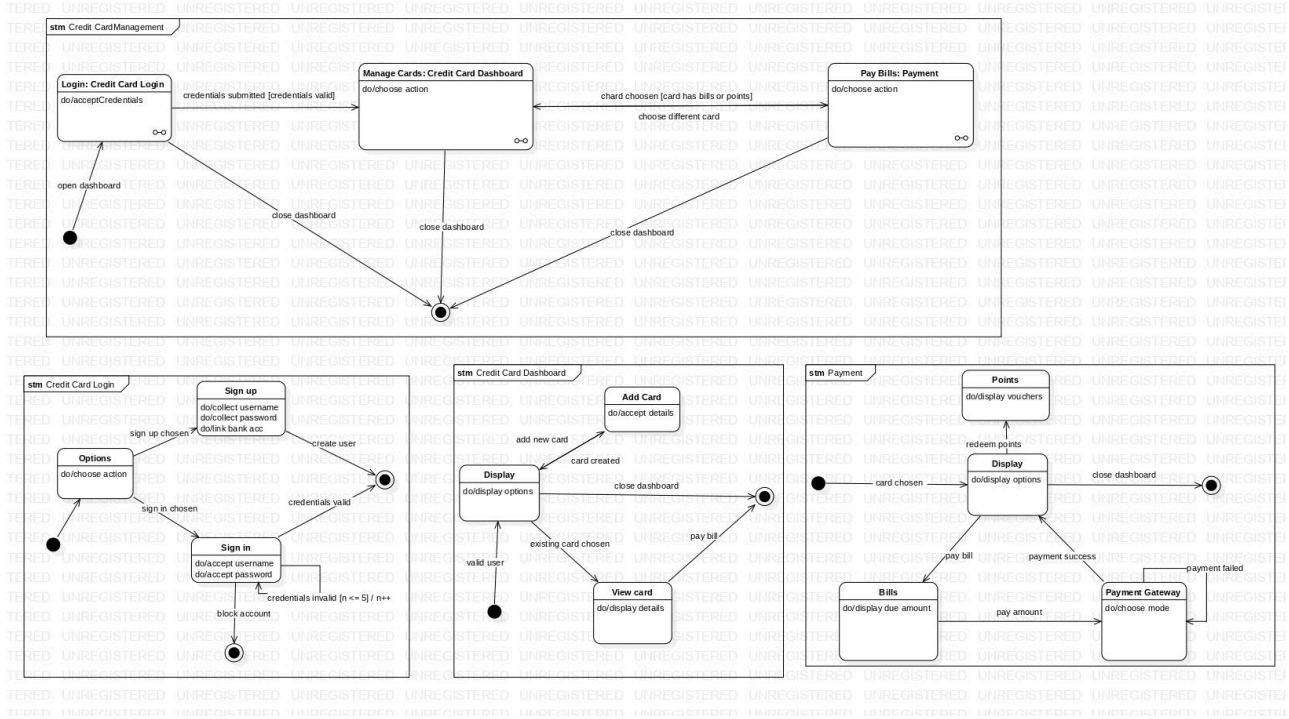
2.4.1 Simple State Diagram



Description:

- States include Off, Idle, Maintenance, Out of Service, and Serving Customer.
- Captures activities like powering on/off, card authentication, PIN validation, and transaction processing.
- Includes maintenance and error scenarios, such as invalid cards or failed PIN validation.
- Focuses on ensuring smooth transaction flow and handling failures or service disruptions.

2.4.2 Advanced State Diagram

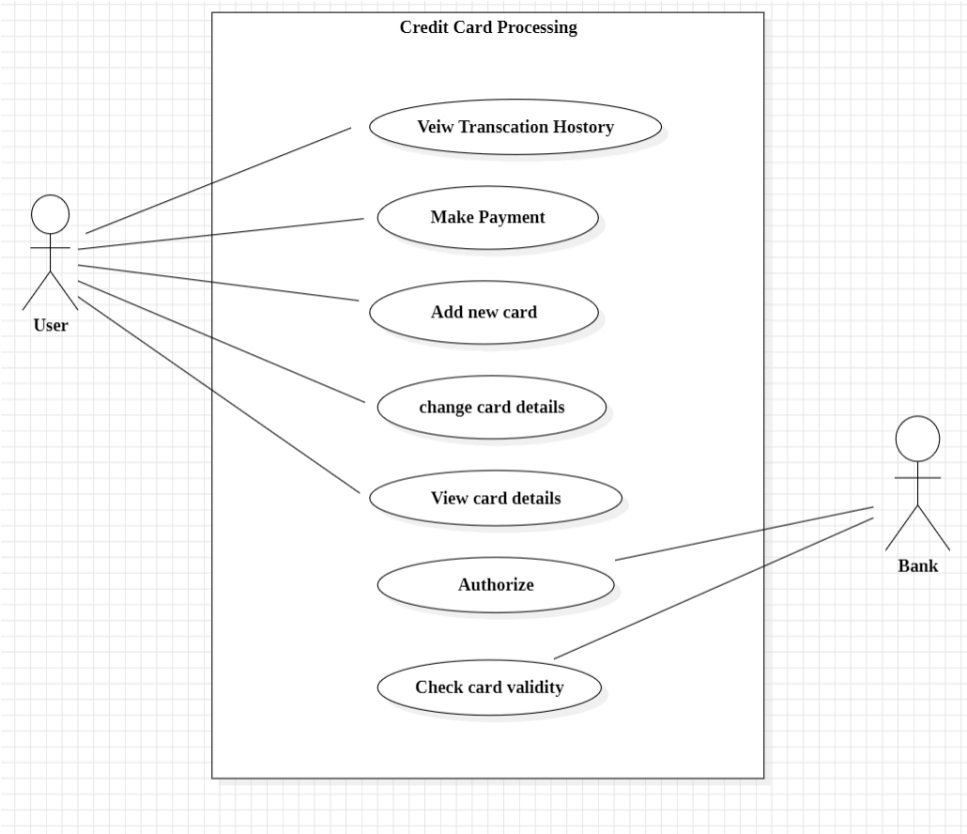


Description:

- The diagram captures user actions such as signing up, signing in, managing cards, paying bills, and redeeming points.
- States include Login, Credit Card Dashboard, and Payment.
- Transitions occur based on user inputs, like entering valid credentials, adding a card, or completing a payment.
- Highlights error handling (e.g., invalid login attempts and blocking accounts after a certain limit).

2.5 Use case diagram

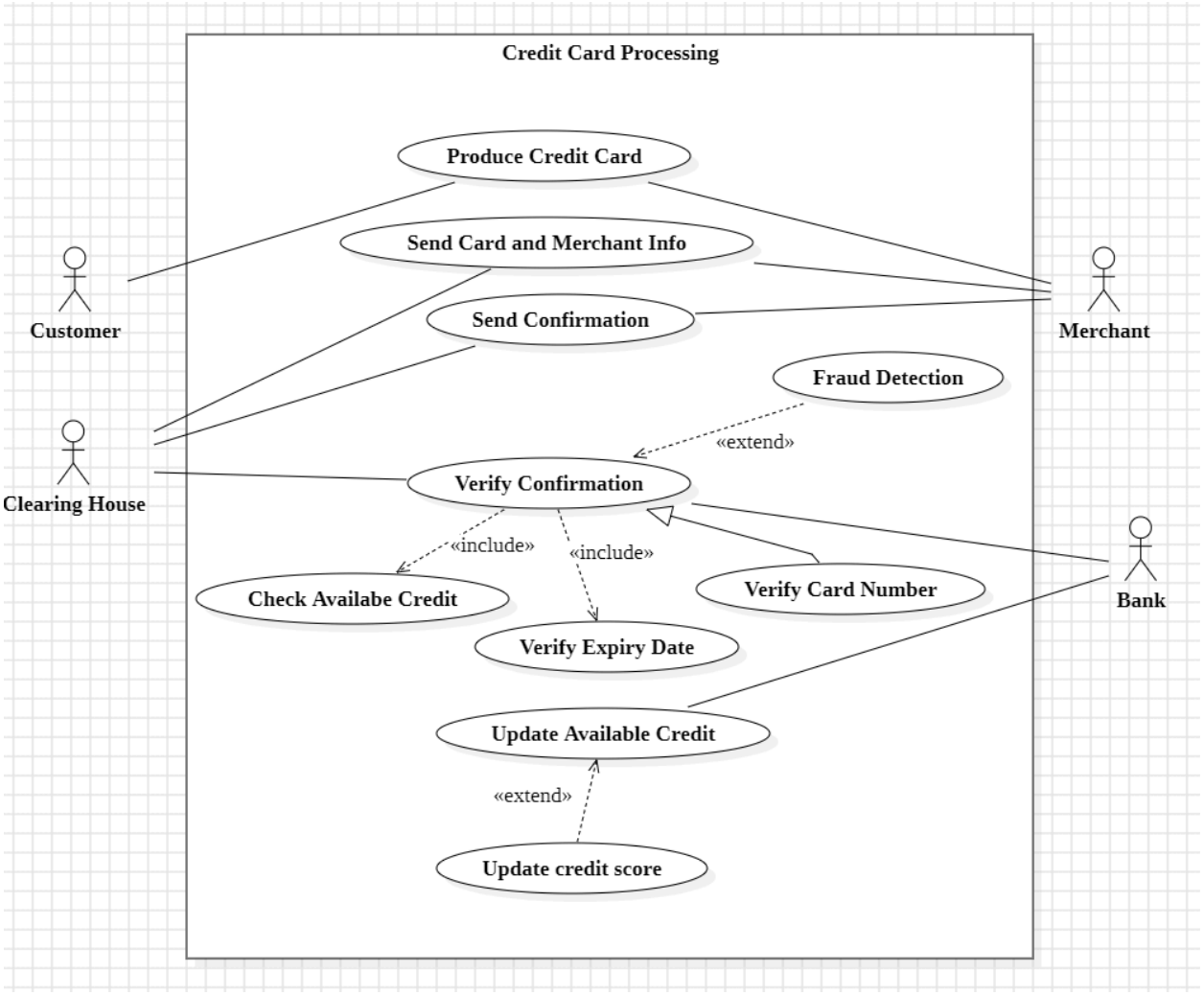
2.5.1 Simple Use Case Diagram



Description: This diagram shows the primary use cases for a credit card processing system, focusing on the interaction between users (e.g., User and Bank) and the system.
Key Use Cases:

- View Transaction History: Allows the user to review past transactions.
- Make Payment: Facilitates bill payments via the credit card system.
- Add New Card: Enables users to add a new credit card to their profile.
- Change Card Details: Allows modifications to existing credit card details.
- View Card Details: Lets users access information about their cards.
- Authorize: Interaction with the bank to approve transactions.
- Check Card Validity: Validates the card's status with the bank.

2.5.2 Advanced Use Case Diagram

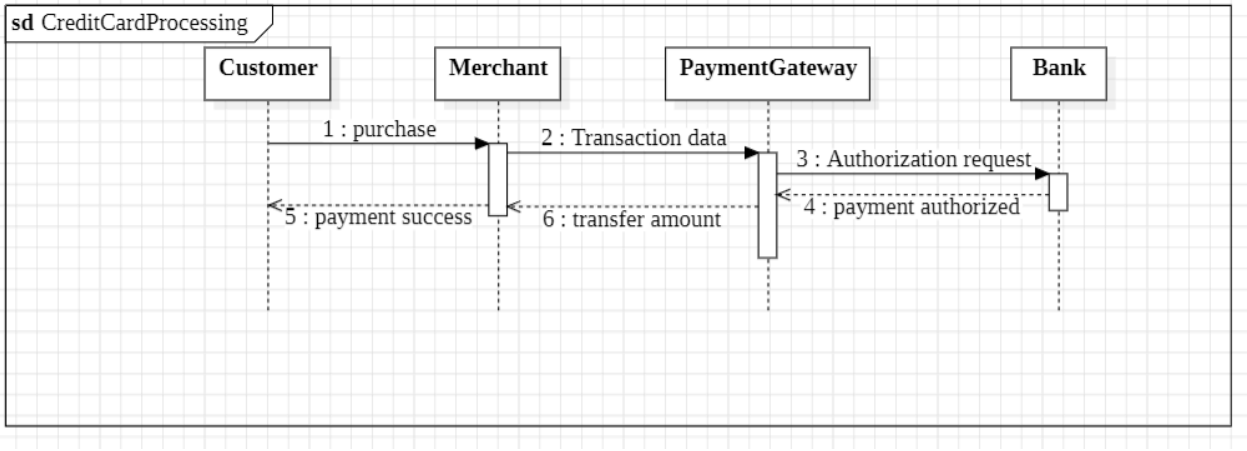


Description: The advanced use case diagram expands on the simple version by providing a more detailed and nuanced view of the credit card processing system. It introduces additional use cases, such as:

- Customer Authentication: Ensures the user is authorized to access and use the credit card.
- Fraud Prevention: Detects and mitigates fraudulent activities during transactions.
- Enhanced Verification: Includes steps like advanced card number checks, dynamic credit validation, and expiry verification.
- Communication with External Systems: Highlights interactions with the Clearing House and Bank to validate transactions and update the system.

2.6 Sequence Diagram

2.6.1 Simple Sequence Diagram

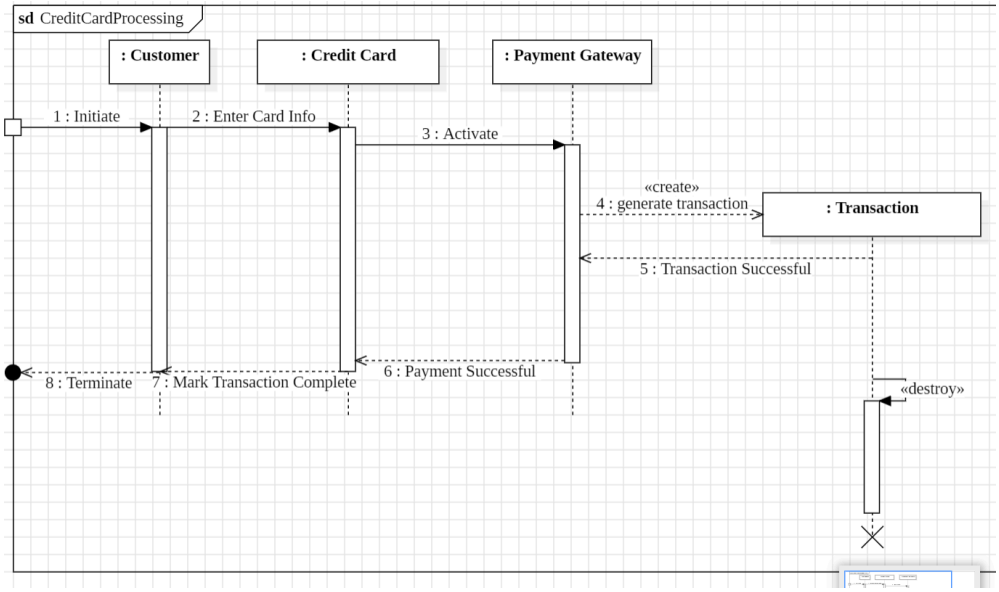


Description: This diagram represents the sequence of interactions among the system components during a credit card transaction.

Steps:

1. Customer initiates a purchase request with the Merchant.
2. The Merchant sends transaction data to the Payment Gateway.
3. The Payment Gateway requests authorization from the Bank.
4. The Bank validates the transaction and sends back authorization.
5. Payment Gateway notifies the Merchant of a successful payment.
6. The Merchant confirms the purchase to the Customer and transfers the amount to the bank.

2.6.2 Advanced Sequence Diagram

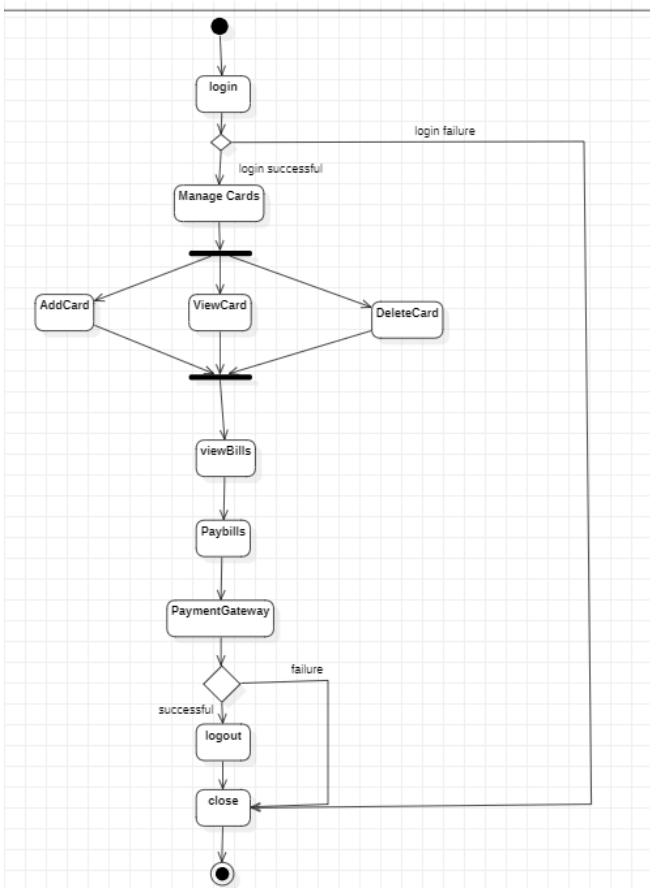


Description:

1. Customer Initiates Transaction: The process starts when the customer enters their credit card details.
2. Authentication & Validation: The system authenticates the customer's credentials and validates the credit card details, including expiry date and available balance.
3. Fraud Detection Check: The system performs fraud detection by analyzing patterns or using external fraud detection services.
4. Generate Transaction: Once validated, the payment gateway creates a transaction object and sends it for processing.
5. External Communication: The payment gateway interacts with the bank or clearing house to authorize the payment.
6. Transaction Completion: Upon approval, the system marks the transaction as successful and notifies the customer and merchant.

2.7 Activity Diagram

2.7.1 Simple Activity Diagram

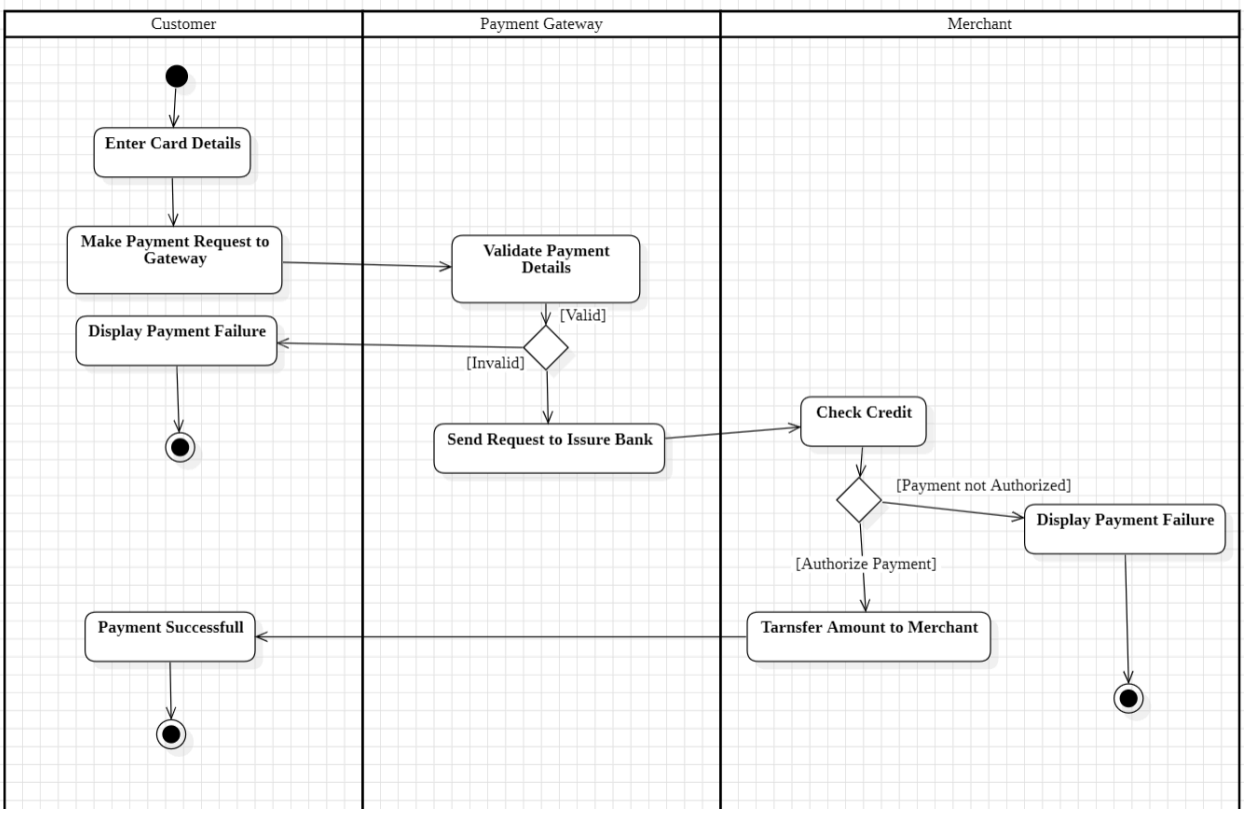


Description: This diagram illustrates the flow of activities in the credit card processing system, from logging in to managing cards and paying bills.

Key Activities:

- Login: Starts the process with either successful authentication or failure.
- Manage Cards: Includes adding, viewing, or deleting credit cards.
- View Bills and Pay Bills: Users can check outstanding bills and make payments.
- Payment Gateway: Handles the payment process, with success leading to logout and failure prompting a retry.
- Logout: Ends the session, closing the interaction with the system.

2.7.2 Advanced Activity Diagram



Description: This activity diagram containing swimlanes for Customer, Payment Gateway and Merchant, illustrates the payment process in the credit card system:

- Customer: Enters card details and makes a payment request to the payment gateway.
- Payment Gateway: Validates the card details and sends a request to the issuing bank.
- Merchant : Checks the credit status of the customer.
    - If payment is authorized, transfer the amount to the merchant.
    - If payment fails (invalid card details or insufficient credit), failure is displayed to the customer.
- Outcome:
    - Payment success: Funds are transferred to the merchant.
    - Payment failure: Relevant failure message is shown to the customer.

Library Management System

3.1 Problem Statement

Develop a software solution to streamline library management processes, incorporating both manual operations by staff and an automated online platform for book borrowing, returning, and inventory management. The system will maintain a centralized database to track book availability, user profiles, borrowing history, and payment details, while supporting real-time updates and secure authentication. Users will be able to browse the catalog, reserve books, renew loans, and make payments online, ensuring seamless integration with the physical library operations. The system must handle multiple transactions simultaneously, enforce borrowing limits, and support notifications for due dates and overdue returns. Staff will manage book inventories, maintain records, and oversee security, while the centralized platform will enable efficient reporting, analytics, and support for subscription-based membership plans.

3.2 Software Requirements Specification (SRS) Document

1. Introduction

1.1 Purpose of this Document: To specify the software requirements of a Library Management System.

1.2 Scope of this Document

- Users are allowed to borrow a book post 9am and return a book before 9pm.
- Allowed to borrow only 3 books at a time and should always produce a library card while borrowing.

1.3 Overview: Detailed information about the requirements and general borrowing procedures.

2. General Description

2.1 Product Perspective

- Library Staff Perspective:

    - They can view the details of the book being borrowed and the customer details borrowing the book.
    - Complete information about the working staff like name, working details, etc.
- Book Perspective:
    - All information about the available books, such as ID, issue date, return date, author, publish date, and the customer who borrowed the book.
- Customer Perspective:
    - Details of all available books, information on books, working hours of the library, and subscription plans.

2.2 Product Functions

- Borrowing books
- Returning books
- Donating books
- Security System
- Library Staff Maintenance

2.3 User Characteristics

- Customers must have a valid library card to borrow/return books.

2.4 General Constraints

- Customers must have a valid library card to borrow/return books.
- Books should be borrowed after 9am and returned before 9pm.
- Library cards can be obtained through subscription plans only.
- Customers can borrow only 3 books at a time.
- Books can be borrowed only for 3 weeks.

2.5 Assumptions and Dependencies

- Customers must ensure that the borrowed books are returned in original condition.
- Donated books must be in good condition.
- Library cards can be obtained only if there is a valid subscription.

3. Functional Requirements

3.1 Book Borrowing and Returning System

- Books can be borrowed by scanning the library card.

3.2 Library Card Generating System

- Library cards will be provided only with valid proof of subscription to the library.

4. Interface Requirements

- User Interface : Borrow/Return Functionality
- API
- Database Interface: For books and customer details.
- User Authentication
- For generating library cards.

5. Performance Requirements

- Response Time: The system should return search results for books within 2 seconds for a typical query.
- Availability: The system should be available to the users at all times.
- Error Handling: Handle errors efficiently.
- Security: All information should be stored safely.

6. Design Constraints

1. The database should be large enough to store all the information about customers, staff, and book details and should be a relational database.
2. The system should be developed in the form of microservices and should be containerized.

7. Non-Functional Attributes

- Security
- Reliability

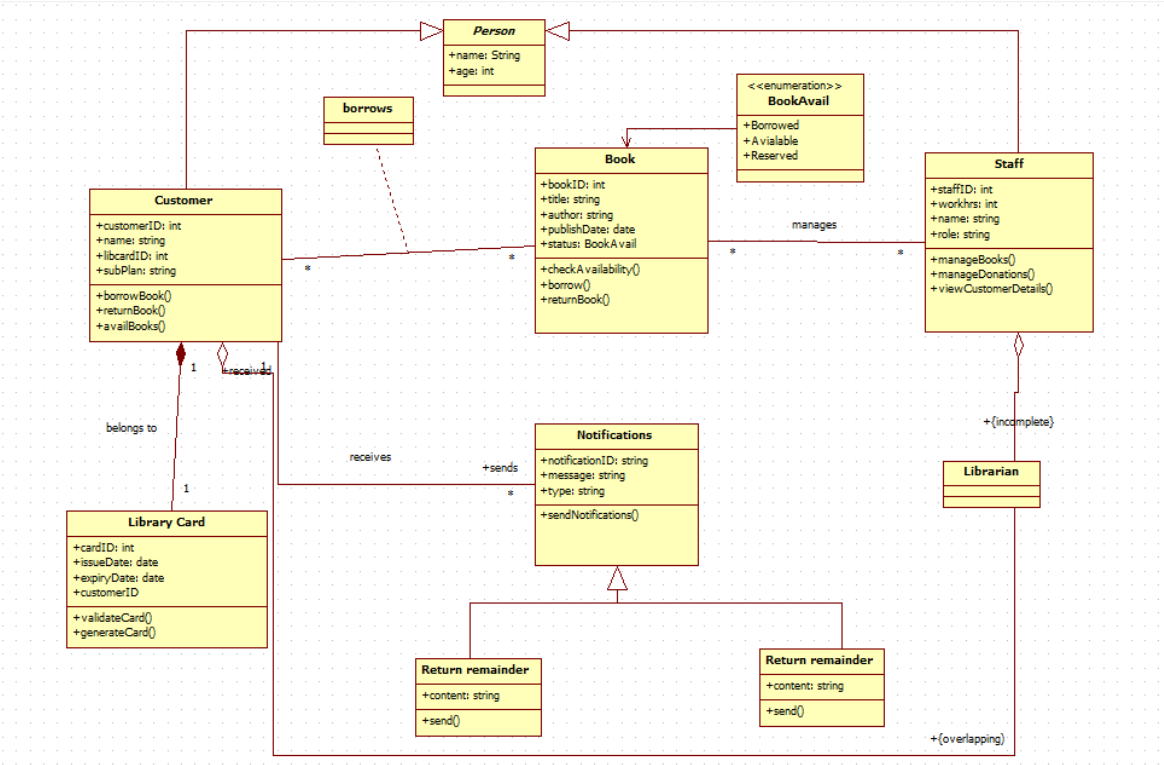
- Reusability
- Application Compatibility
- Data Integrity
- Availability
- Capacity

8. Preliminary Schedule and Budget

- Schedule: In this, the initial version and budget of the project plan are explained, which include the overall time duration required and the overall cost required for development of the project.

3.3 Class Diagram

3.3.1 Advanced Class Diagram



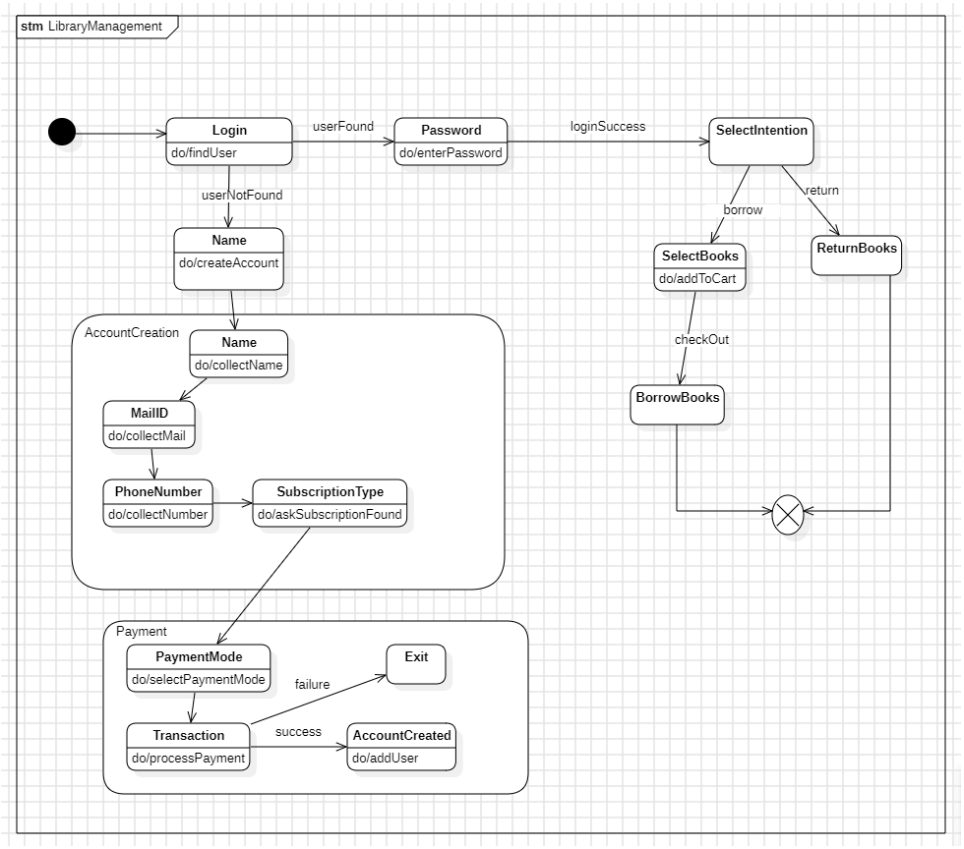
Description:

This expanded class diagram includes more explicit connections between entities:

- Borrow Relationship: Connects customers and books, detailing the process flow.
- Notifications System: Tracks and sends messages regarding books or due dates.
- Return Reminders: Specializes in follow-ups for borrowed books, ensuring timely returns.

3.4 State Diagram

3.4.1 Simple State Diagram

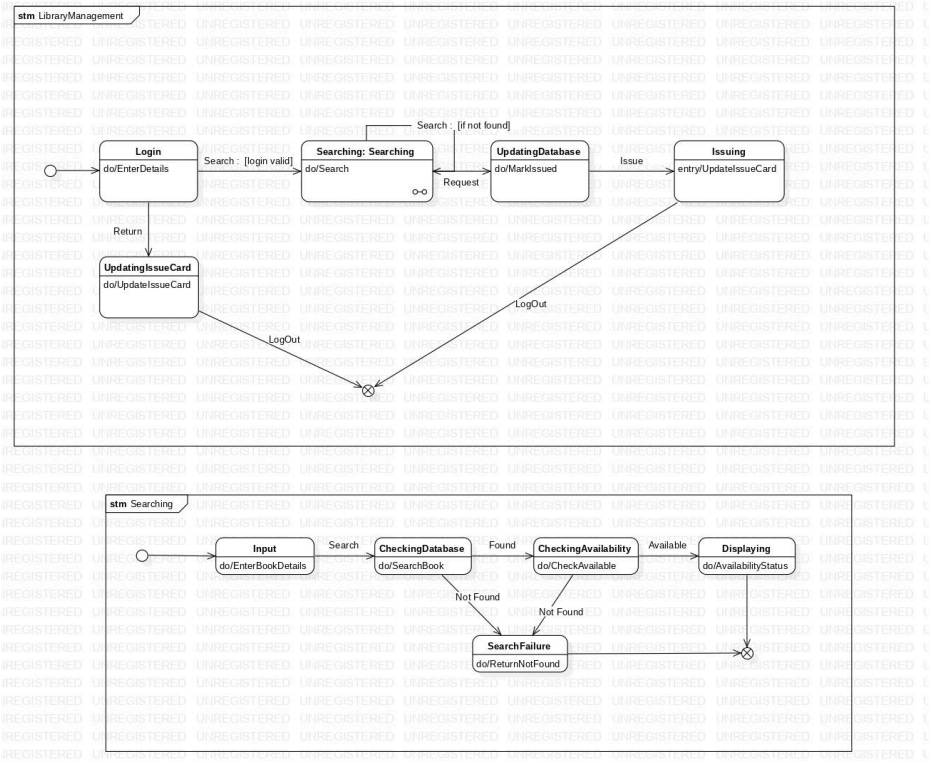


Description: This diagram describes the various states and transitions within a library management system. Key highlights:

- Login Process: Users provide credentials to enter the system.
- Searching State: Users can search for books, which involves inputting details, checking availability, and displaying results.
- Issuing State: If a book is found and requested, the system updates the database and issues the book.

- Updating Issue Card: Maintains the record of issued books for each user.
- Logout Process: Indicates the user's exit from the system. Sub-state diagrams (e.g., Searching) further detail inner workflows like inputting book details, checking the database, and handling search failures.

3.4.2 Advanced State Diagram

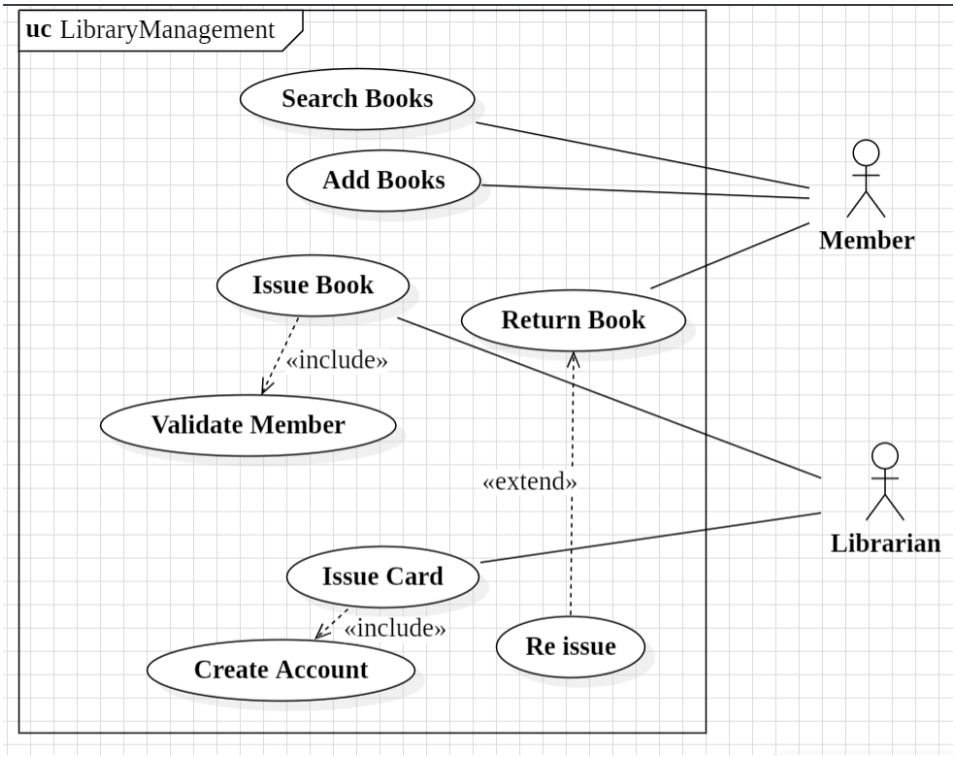


Description: This diagram elaborates on additional functionalities like:

- Account Creation: For new users, the system collects personal details (name, email, phone number) and subscription type before account activation.
- Transaction Handling: Payment modes and subscription processing are covered here, with success leading to account creation.
- Book Management: After logging in, users can select intentions like borrowing or returning books, with distinct workflows for both actions.

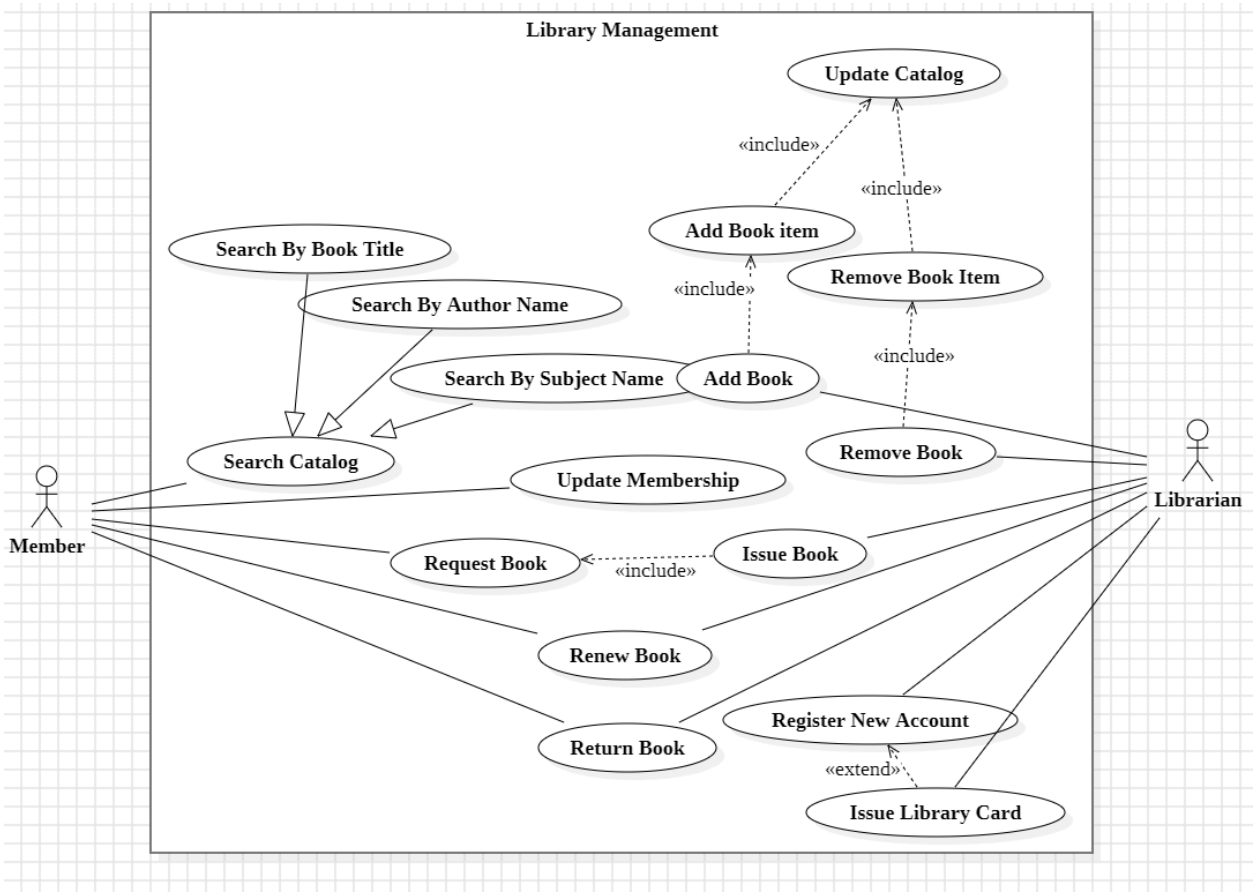
3.5 Use case diagram

3.5.1 Simple Use Case Diagram



Description: The use case diagram represents the functionalities of a library management system. It highlights the interactions between two primary actors, Members and Librarians, and the system. Members can search for books, return books, and request book issuance, which includes validating membership. Librarians manage tasks such as adding books, issuing cards, and reissuing books, which extends the book-return functionality. The diagram uses relationships like <<include>> and <<extend>> to depict dependencies between processes, ensuring clarity in the flow of operations.

3.5.2 Advanced Use Case Diagram

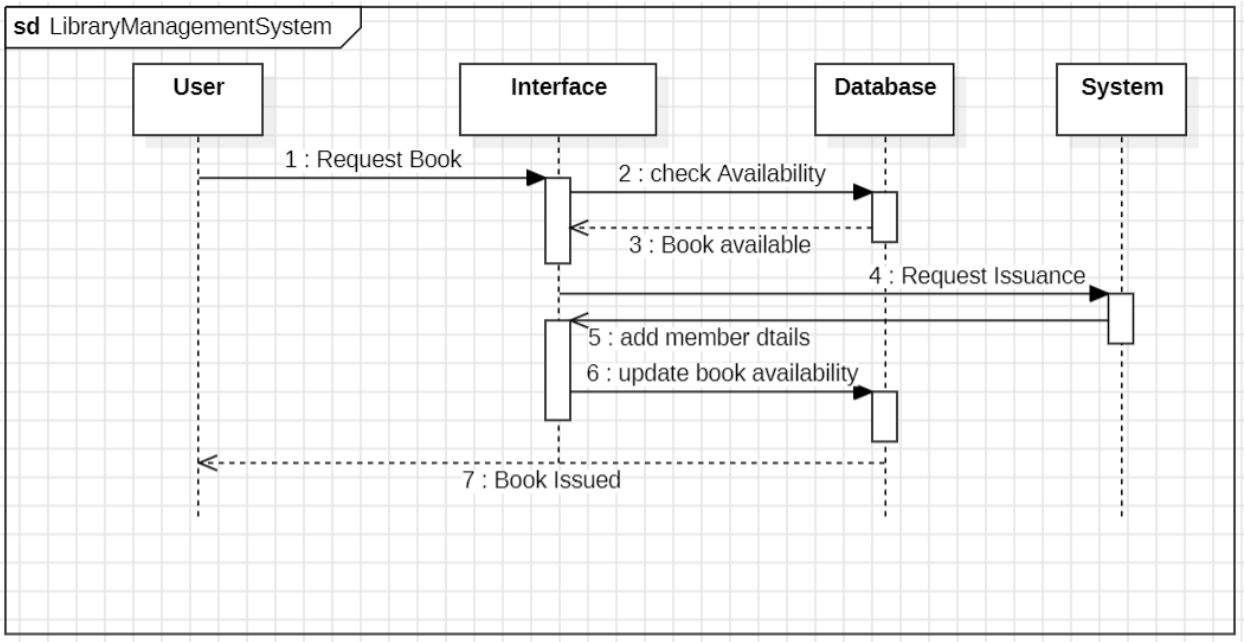


Description: The use case diagram models the interactions between users (actors) and the Library Management System.

- Actors: The main actors are the Member and Librarian.
- Use Cases: It includes actions such as searching the catalog (by title, author, or subject), requesting books, returning books, and updating membership for members. For librarians, it includes issuing books, updating the catalog (adding/removing books), and managing user accounts.
- Relationships: Dependencies like include and extend define additional functionalities and shared responsibilities between use cases.

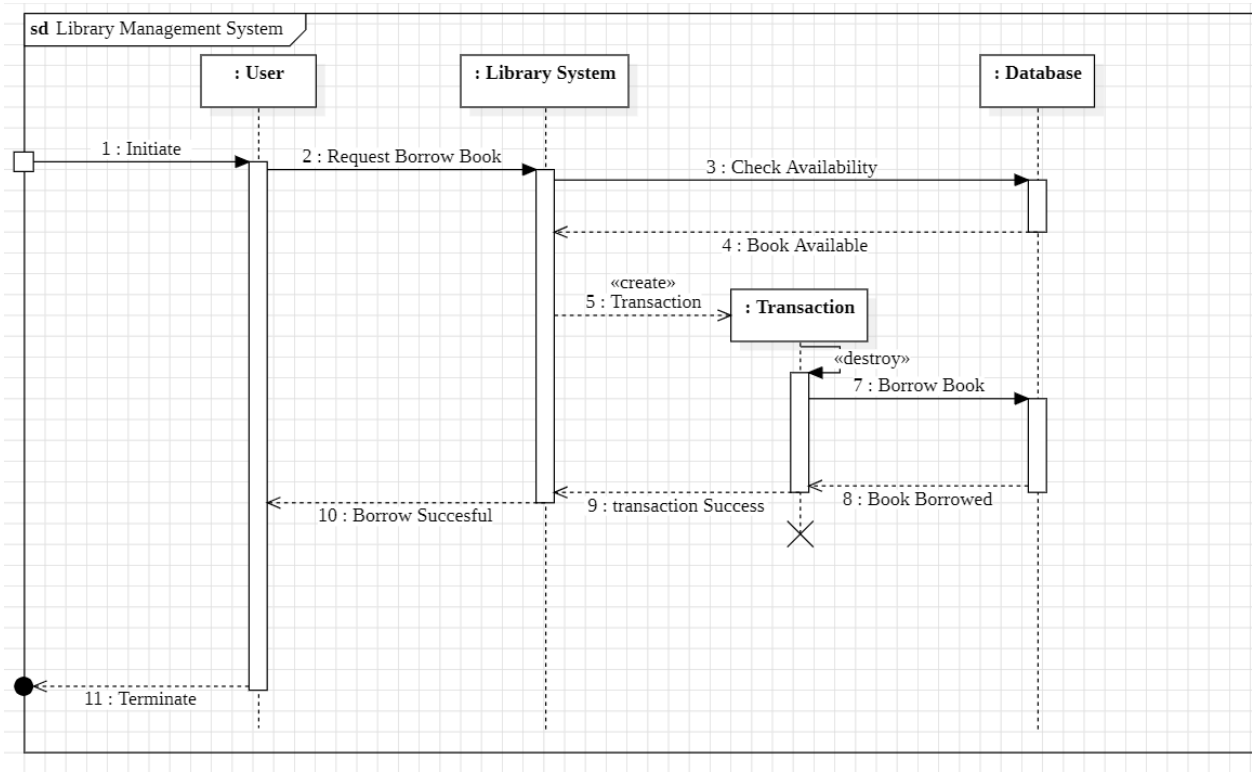
3.6 Sequence Diagram

3.6.1 Simple Sequence Diagram



Description: The sequence diagram illustrates the dynamic interaction among components of the library management system during a book issuance process. The flow begins with the user requesting a book through the interface. The system checks the book's availability in the database, and if available, the issuance request is processed. Member details are added, the database is updated to reflect the new status, and the book is issued to the user. This step-by-step flow highlights the real-time communication between the user, interface, database, and system components.

3.6.2 Advanced Sequence Diagram

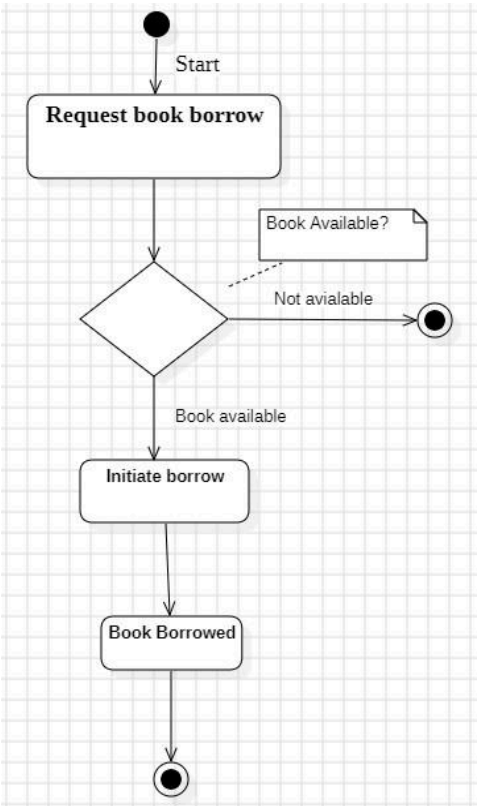


Description: This diagram illustrates the sequence of interactions during the book borrowing process.

- Lifelines: The key participants are the User, Library System, Database, and a temporary Transaction object.
- Flow: The user initiates a book borrow request, which is processed by the library system by checking the database for availability. If the book is available, a transaction is created, the book is issued, and the transaction ends.
- Messages: Shows synchronous and asynchronous messages exchanged among components, ensuring smooth operation.

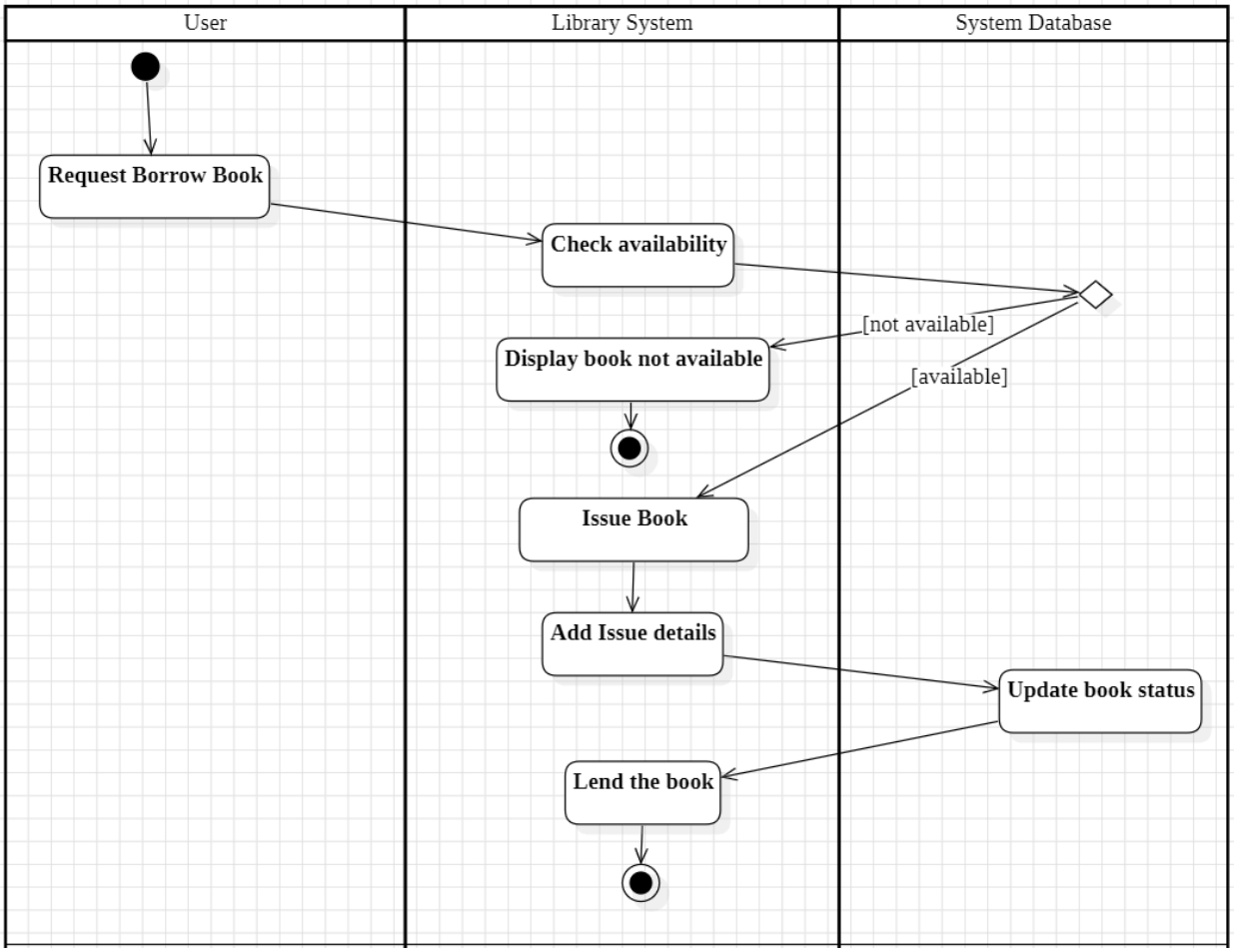
3.7 Activity Diagram

3.7.1 Simple Activity Diagram



Description: The activity diagram outlines the workflow of borrowing a book from the library. It starts with the user requesting to borrow a book. A decision node checks the book's availability. If the book is not available, the process ends; otherwise, the borrowing process is initiated. Upon successful initiation, the book is borrowed, and the process concludes. This diagram emphasizes the conditional logic and sequential steps involved in the borrowing process.

3.7.2 Advanced Activity Diagram



Description: The activity diagram represents the workflow for borrowing a book from the library.

- Flow: The process starts when a user requests to borrow a book.
- Steps: The system checks the availability of the book in the database. If unavailable, a "book not available" message is displayed. If available, the book is issued, details are recorded, and the book is handed to the user.
- Swimlanes: The activities are divided among three entities: User, Library System, and System Database, showing their responsibilities.

Stock Maintenance System

4.1 Problem Statement

The Stock Maintenance System is designed to efficiently manage and track inventory levels, orders, sales, and deliveries, providing functionalities such as adding, updating, and removing stock items, generating inventory and sales reports, managing suppliers, and issuing low-stock notifications. Catering to inventory staff, managers, and suppliers, the system ensures real-time stock updates, intuitive interfaces, and role-based access control. Key features include automated low-stock alerts, analytical dashboards, supplier management, and seamless integration with existing ERP systems. Developed using a microservices architecture, the system emphasizes scalability, usability, and security, supporting at least 10,000 stock items while ensuring data encryption and rapid processing of updates within 2 seconds. With a projected timeline of 5 months and a budget of ₹35 lakhs, the project covers all stages from design to deployment, ensuring a robust and user-friendly solution for effective inventory management.

4.2 Software Requirements Specification (SRS) Document

1. Introduction

1.1 Purpose of this Document: To specify the software requirements for a Stock Maintenance System, which helps manage and track inventory levels, orders, sales, and deliveries.

1.2 Scope of this Document: Users can add, update, and remove stock items, track stock levels, generate reports, and manage suppliers. The system supports notifications for low stock levels and facilitates order management.

1.3 Overview: This document provides detailed information about the requirements and general procedures for stock management.

2. General Description

2.1 Product Perspective:

- Staff Perspective:

    - View details of stock items, including quantities, descriptions, prices, and suppliers.
    - Generate and view reports on stock levels, sales, and orders.
- Management Perspective:
    - Access analytical dashboards for inventory trends, low stock alerts, and financial summaries.
    - Manage supplier details and performance.

2.2 Product Functions:

- Add, update, and delete stock items
- Track stock levels and manage reordering
- Generate reports (sales, inventory status, supplier performance)
- Manage suppliers and purchase orders
- Receive notifications for low stock levels

2.3 User Characteristics:

- Users include inventory staff, managers, and suppliers.
- Staff must have basic computer skills to operate the system.

2.4 General Constraints:

- Users must have appropriate permissions to add or modify stock.
- The system must operate within the organization's existing IT infrastructure.
- Stock updates should occur in real-time to ensure data accuracy.

2.5 Assumptions and Dependencies:

- Users will input accurate stock information.
- The system will rely on existing database infrastructure for data storage.
- Users will have internet access for online features if applicable.

3. Functional Requirements:

- Stock Management System:

    - Users can add new stock items, including SKUs, descriptions, and quantities.
    - Users can update existing stock details and remove stock items as necessary.
- Reporting System: Generate reports on inventory levels, sales data, and supplier information.
- Notification System: Automated alerts for low stock levels and reorder reminders.
- Supplier Management: Add and manage supplier details, including contact information and pricing.

4. Interface Requirements:

- User Interface (UI): **Be intuitive and easy to navigate.**
- API Interface: enables external systems to interact with the stock maintenance system
- Database Interface: responsible for interacting with the underlying database to manage stock and related data

5. Performance Requirements

- Response Time: The system should return stock search results within 2 seconds.
- Availability: The system should be available 99.9% of the time, excluding scheduled maintenance.
- Error Handling: The system should provide user-friendly error messages and recovery options.
- Security: All sensitive data should be encrypted in storage and during transmission.

6. Design Constraints

1. Database Structure: The database must be a relational DBMS to maintain structured inventory data.
2. Microservices Architecture: The system should be developed using microservices for modular functionality and scalability.
3. Integration with Existing Systems: The stock maintenance system must integrate with existing ERP or accounting systems.

4. Hardware Limitations: The system must operate on standard server hardware specified by the organization.

7. Non-Functional Requirements:

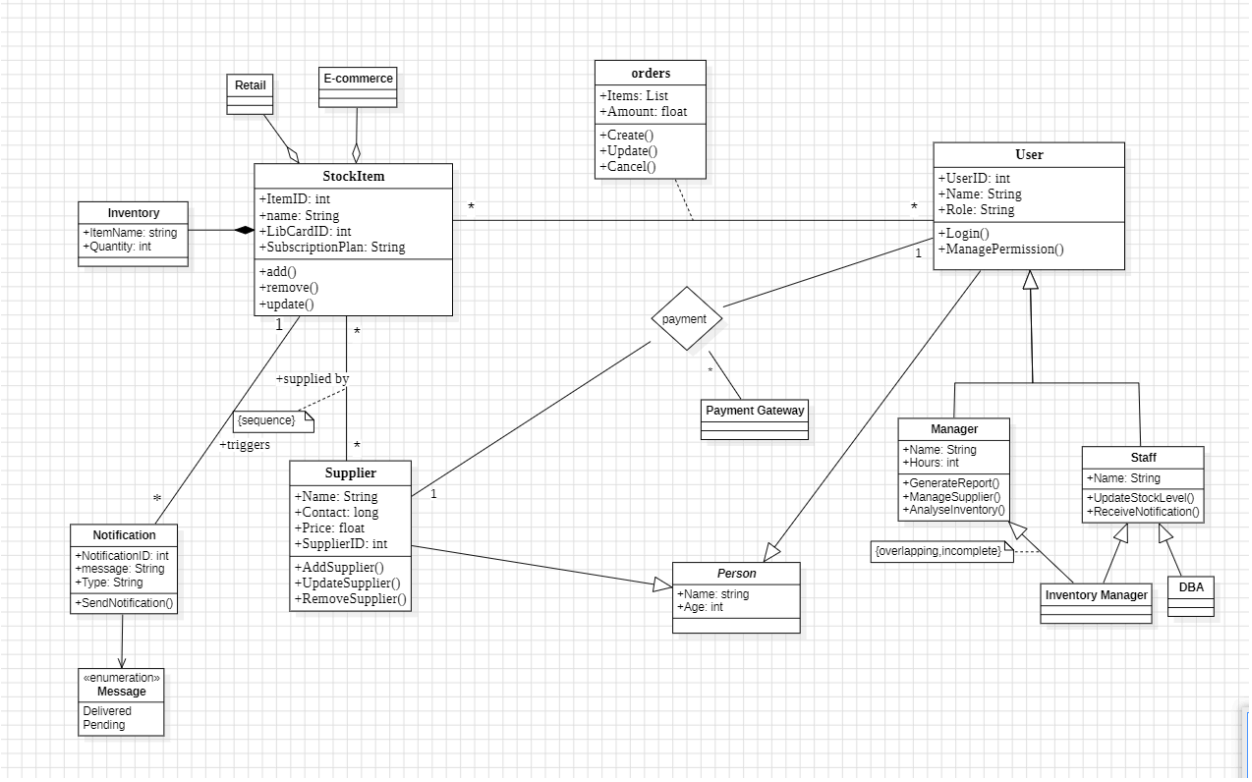
- Performance: The system should process stock updates within 2 seconds.
- Scalability: Capable of handling inventory data for at least 10,000 stock items.
- Security: Ensure confidentiality of sensitive supplier and stock information.
- Usability: Intuitive UI that requires minimal training for new users.

8. Preliminary Schedule and Budget:

The stock maintenance system project is estimated to take approximately 5 months to complete. This includes hiring staff and engineers, planning and designing the software architecture, developing the system, conducting testing and quality assurance, and finally deploying the system with training. The total budget required is estimated to be around ₹35 lakhs, covering personnel costs such as software developers, UI/UX designers, QA testers, and a project manager. It also includes software and database licenses, infrastructure and server costs, training materials, documentation, and miscellaneous expenses like maintenance and contingency.

4.3 Class Diagram

4.3.1 Advanced Class Diagram



Description:

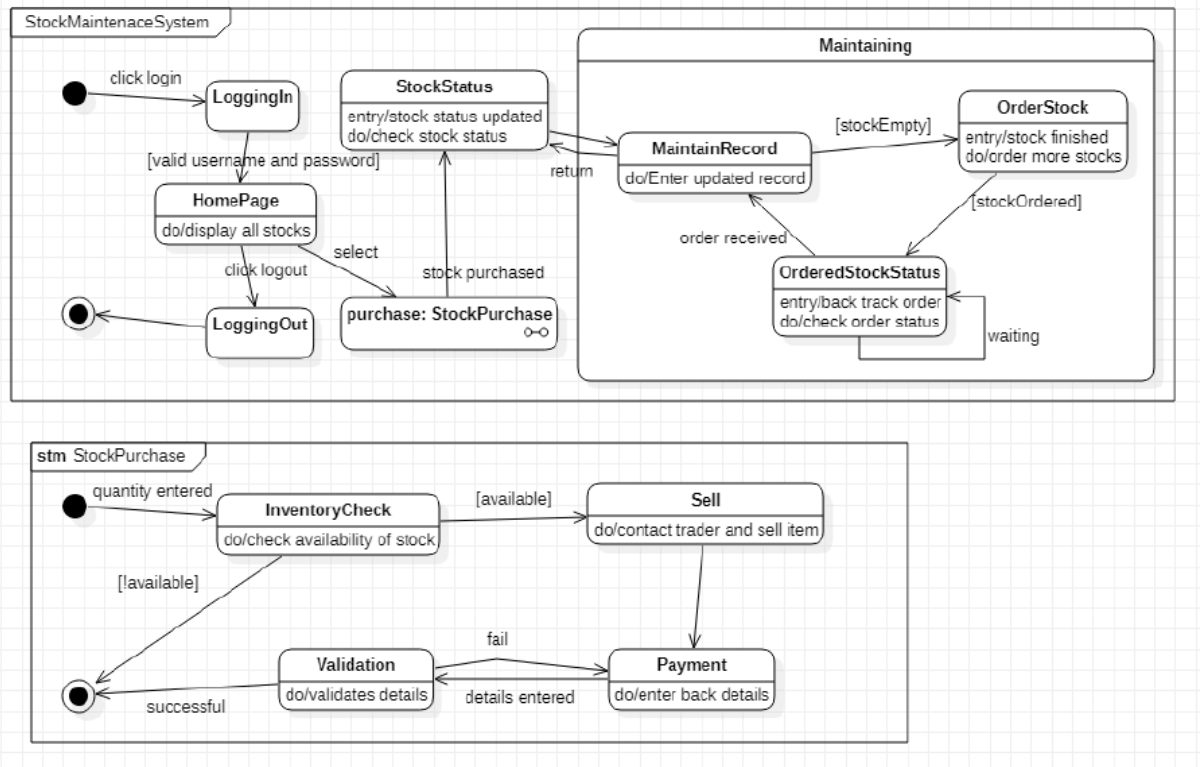
- Classes:
    - StockItem: Represents items in the stock with attributes such as ItemID, name, and SubscriptionPlan. Methods include add(), remove(), and update().
    - Supplier: Tracks suppliers with attributes like Name, Contact, and SupplierID. It includes methods to add, update, or remove a supplier.
    - User: Represents system users with roles such as Manager or Staff, supporting actions like Login() and ManagePermission().
    - Manager and Staff: Specific types of users with additional operations like GenerateReport() for Managers and UpdateStockLevel() for Staff.
    - Notification: Handles notifications in the system, with details such as NotificationID and message.
- Relationships:
    - A StockItem is supplied by multiple Suppliers.
    - Notifications can be triggered by stock events.

- Managers oversee suppliers, and Staff manage stock levels.

- Enumerations:
  - Defines statuses for messages, e.g., Delivered or Pending.

4.4 State Diagram

4.4.1 Simple State Diagram



Description: The diagrams illustrate a Stock Maintenance System and its Stock Purchase subprocess, showcasing how stock levels are managed and purchases are handled.

Stock Maintenance System

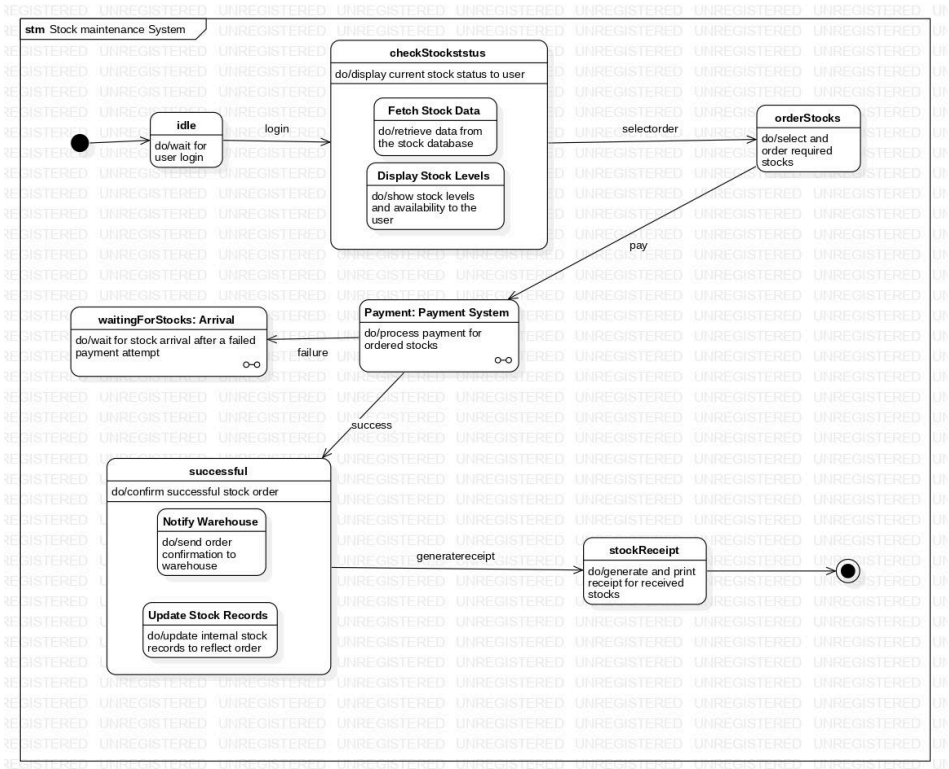
- LoggingIn: Users log in with valid credentials.
- StockStatus: Displays the current stock levels. Users can view stocks, update records, or initiate purchases.
- MaintainRecord: Allows updates to stock records.
- OrderStock: Automatically triggers when stock levels are empty, placing orders for replenishment.

- OrderedStockStatus: Tracks the status of placed orders, including back orders or pending orders.
- LoggingOut: Users log out to end the session.

Stock Purchase

- InventoryCheck: Verifies if the requested stock quantity is available.
- Validation: Checks the entered details; if valid, the process continues.
- Payment: User completes the purchase by entering payment details.
- Sell: Finalizes the transaction by contacting the trader or seller.

4.4.2 Advanced State Diagram



Description:

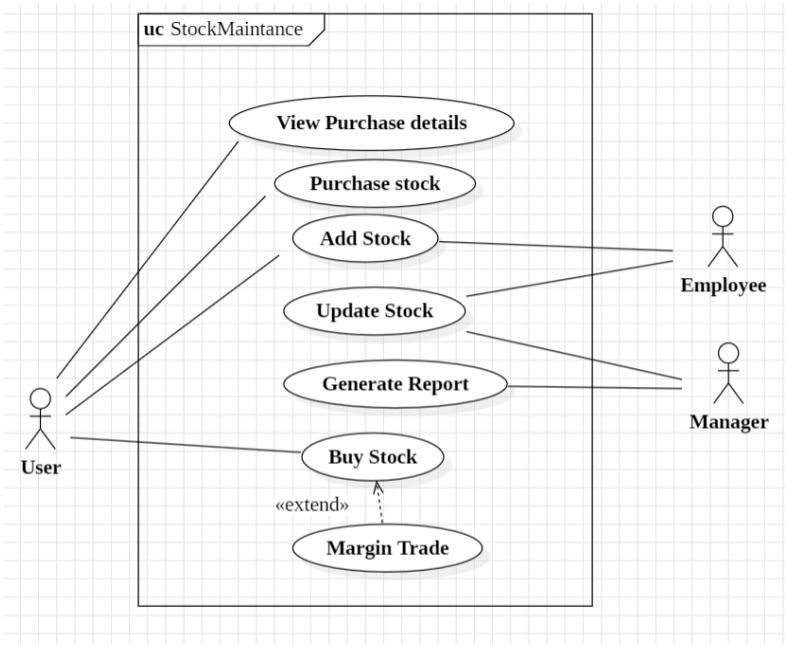
- States:
    - Various conditions a stock item can exist in, such as Available, Out of Stock, or Reserved.
- Transitions:

- Arrows representing the change from one state to another, triggered by events like Add, Remove, or Update.

- Start and End States:
  - Indicates the initialization and final state of a stock item.

4.5 Use case diagram

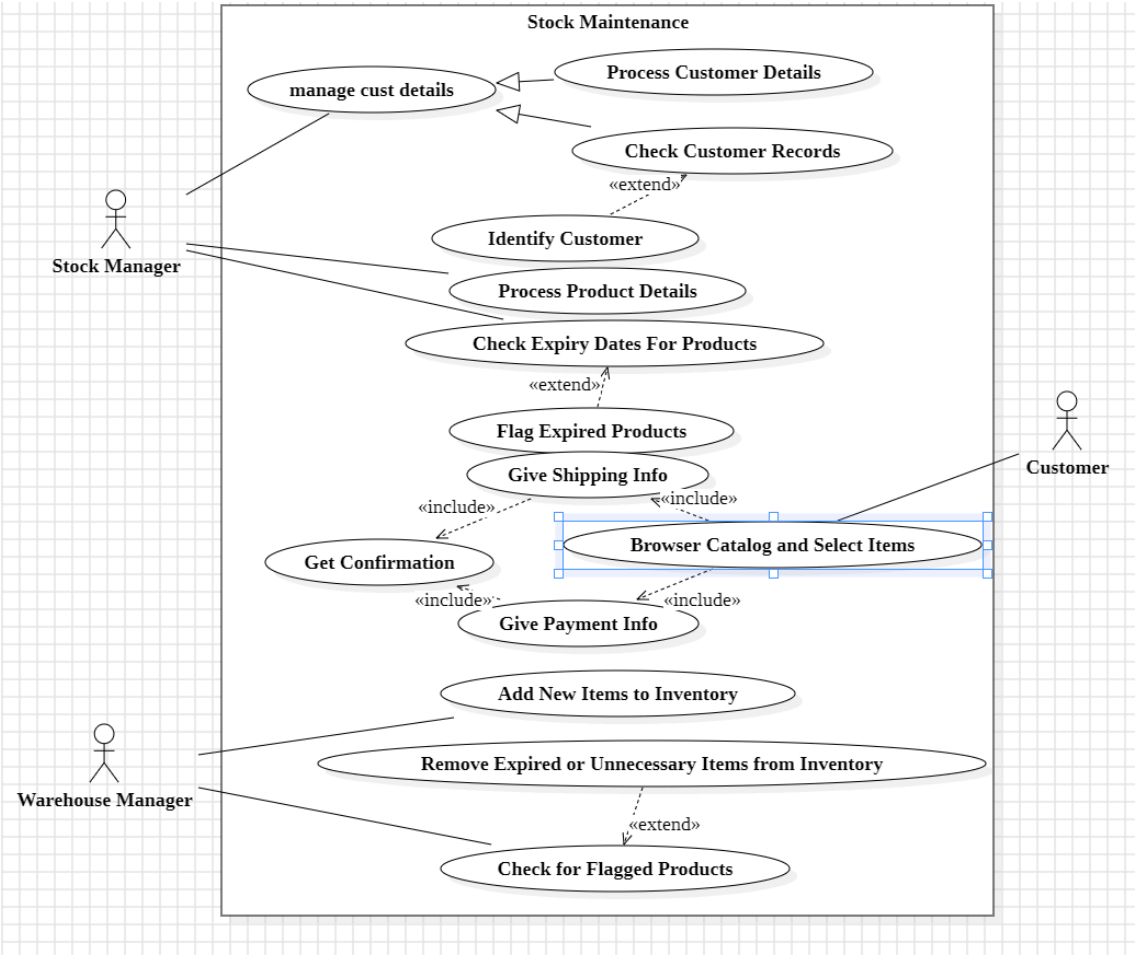
4.5.1 Simple Use Case Diagram



Description: The use case diagram depicts the primary functionalities of the stock maintenance system and the roles involved.

- Actors:
  - User: Can perform actions like viewing purchase details, purchasing stock, adding stock, updating stock, generating reports, and buying stock.
  - Employee: Collaborates with the system for stock-related tasks.
  - Manager: Focuses on generating reports and other managerial actions.
- **Use Cases**:
  - "Buy Stock" extends to "Margin Trade," indicating optional or conditional trading capabilities.
  - Functionalities are connected to the respective roles that interact with them.

4.5.2 Advanced Use Case Diagram

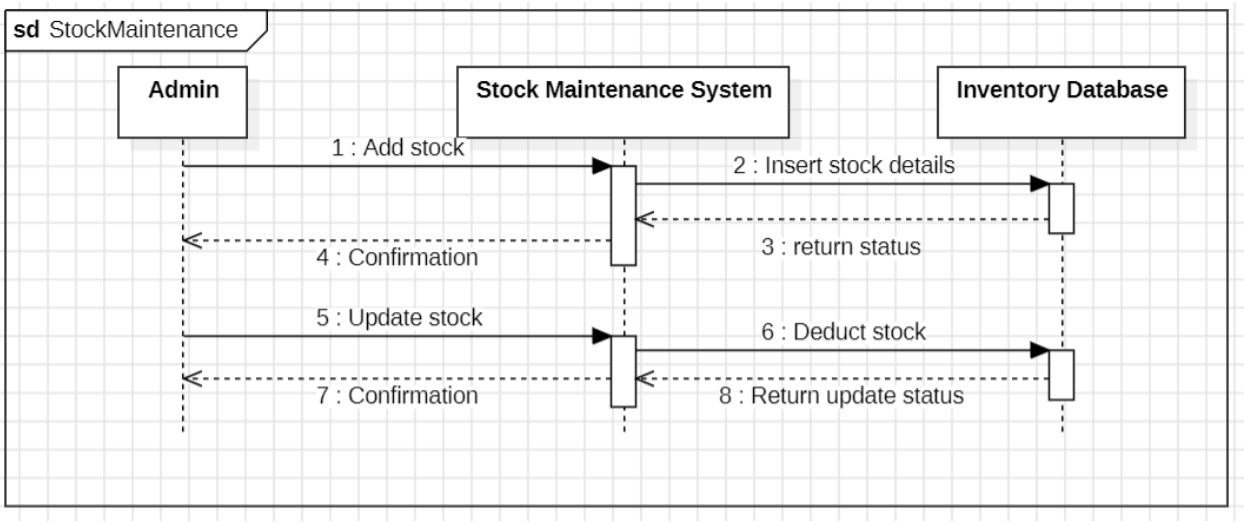


Description: This advanced use case diagram depicts detailed functionalities of the Stock Maintenance system involving Stock Manager, Warehouse Manager, and Customer.

- Stock Manager manages customer details, processes product details, and identifies customers.
- Warehouse Manager handles inventory by adding and removing items and flagging expired products.
- Customer interacts with the system to browse catalogs, give shipping information, and confirm payments.

4.6 Sequence Diagram

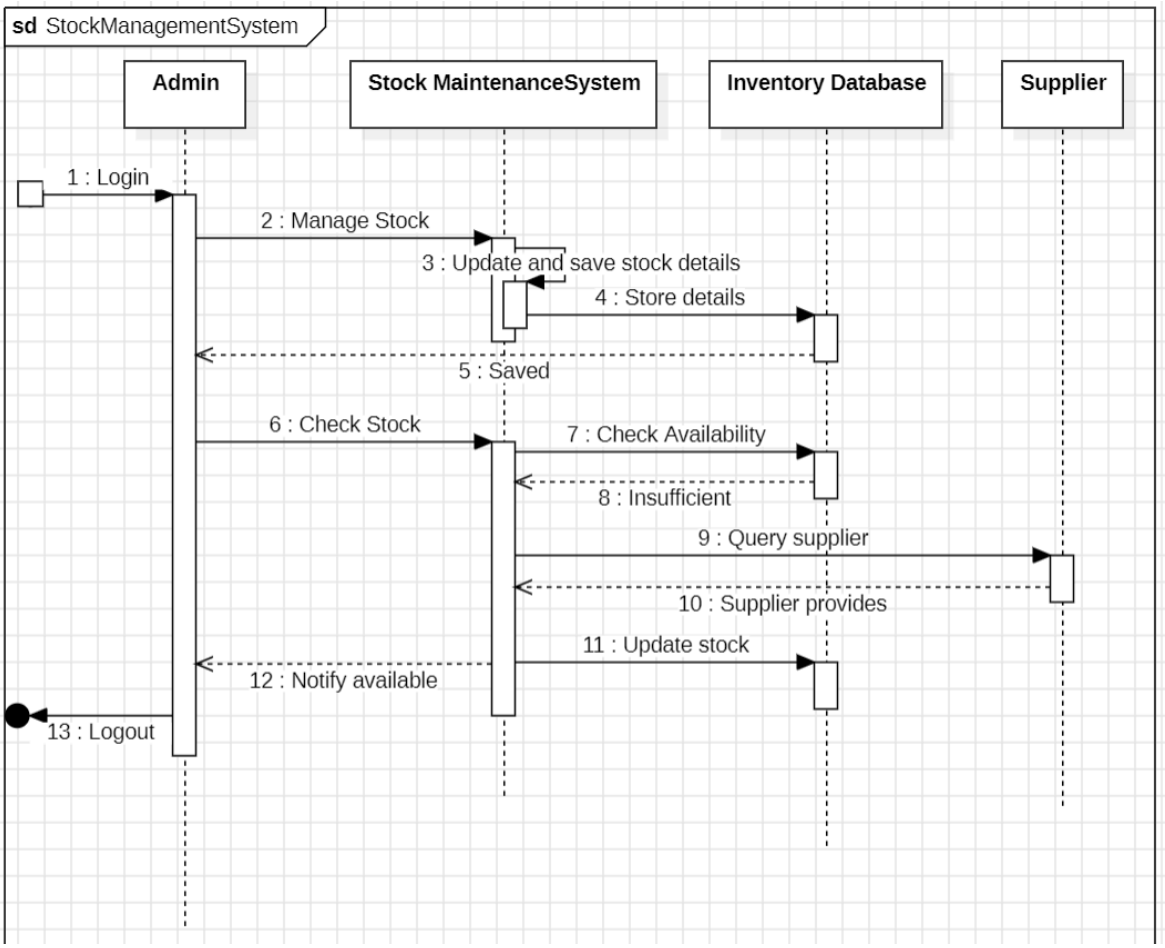
4.6.1 Simple Sequence Diagram



Description: This sequence diagram highlights the flow of interactions between the admin, stock maintenance system, and the inventory database for managing stock updates.

- Flow:
    - Admin adds or updates stock.
    - The stock maintenance system interacts with the inventory database to insert or deduct stock details.
    - Status or confirmation messages are returned at each stage to ensure consistency.

4.6.2 Advanced Sequence Diagram

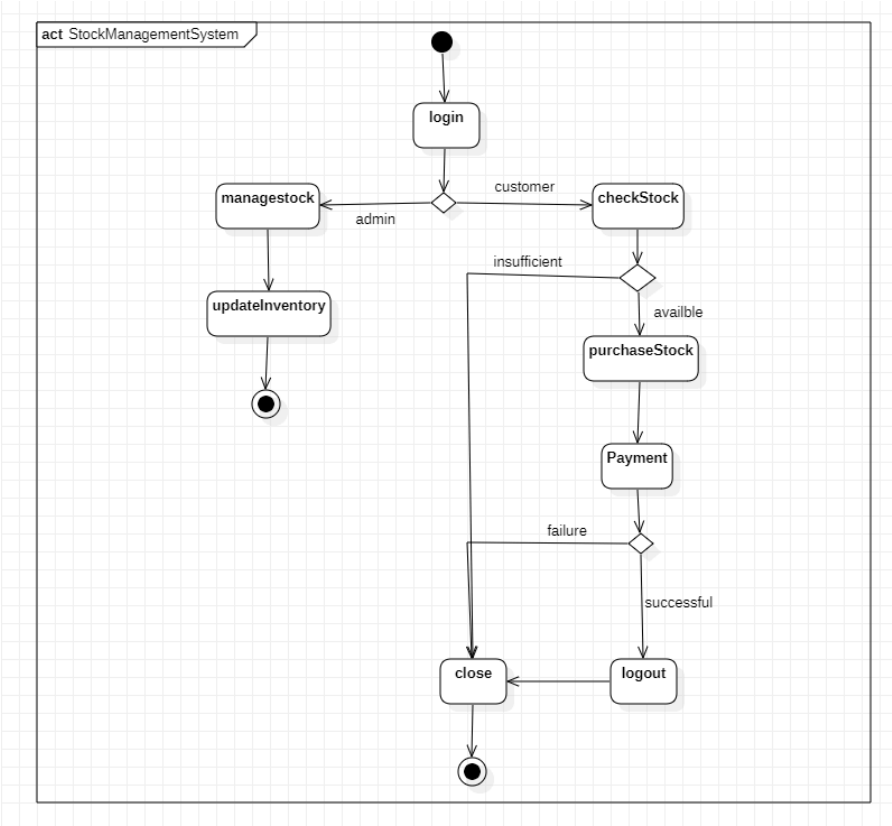


Description: This sequence diagram shows the flow of information among Admin, Stock Maintenance System, Inventory Database, and Supplier.

- The Admin logs in and manages stock, which is updated in the system and saved in the database.
- If stock is insufficient, the system queries the supplier, who provides additional stock.
- Notifications are sent back once the stock is updated.

4.7 Activity Diagram

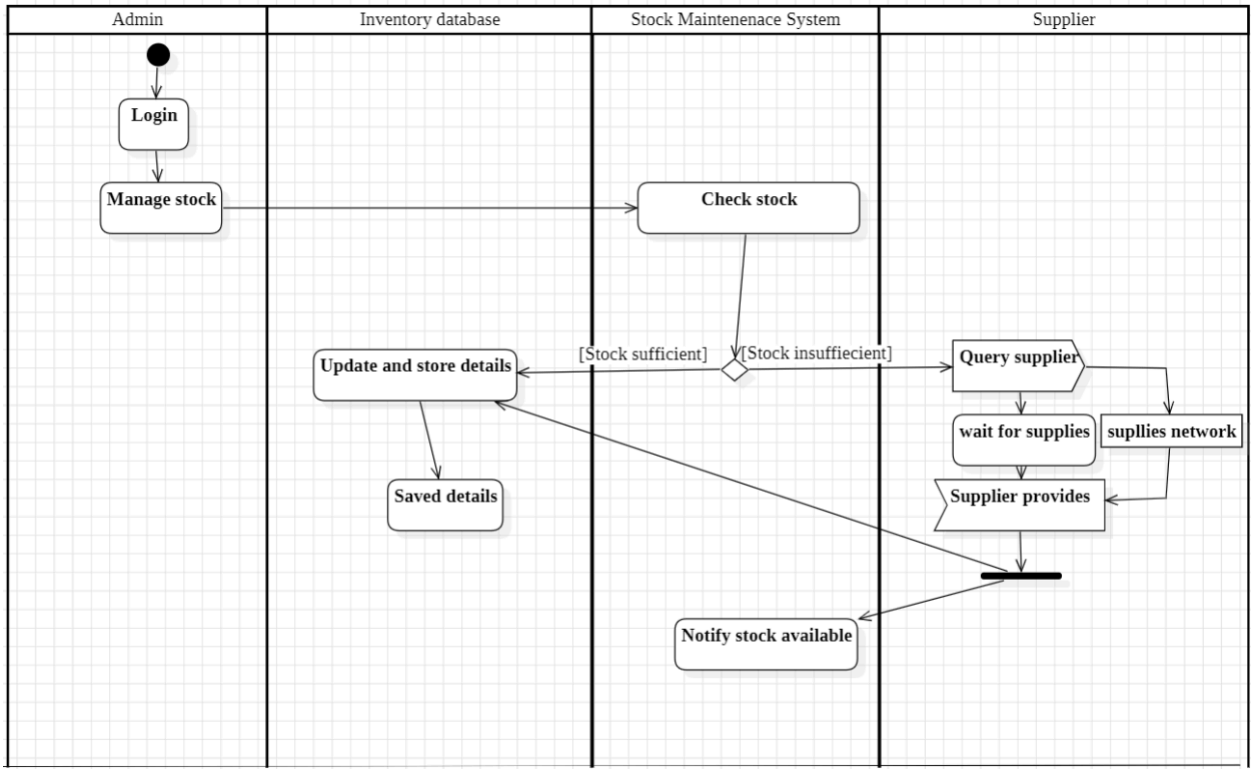
4.7.1 Simple Activity Diagram



Description: The activity diagram outlines the overall process flow of the stock management system.

- Key Activities:
  - Login: Entry point for admin or customer.
  - Manage Stock: Admin manages inventory, leading to inventory updates.
  - Check Stock: Customers check stock availability.
  - Purchase Stock: Customers proceed with purchasing if stock is available, followed by payment.
  - Outcome:
    - Payment failure leads to closure.
    - Successful payment results in logout, completing the process.

4.7.2 Advanced Activity Diagram



Description: This advanced activity diagram highlights the roles of Admin, Inventory Database, Stock Maintenance System, and Supplier in the stock management process.

- The Admin initiates the process by managing stock.
- The Stock Maintenance System checks stock levels and interacts with the supplier if stock is insufficient.
- Stock updates and notifications are handled collaboratively between the system and database.

Passport Automation System

5.1 Problem Statement

Design a Passport Automation System to streamline the end-to-end process of passport application, verification, and issuance. The system must enable applicants to submit applications online, upload documents, track application status, and schedule appointments for document verification and biometric data submission.

Staff should be able to verify applications, manage appointments, and process passport issuance efficiently. The system should ensure real-time updates, secure data storage, and integration with government identity verification systems and postal services for passport delivery.

The solution must comply with legal requirements, support thousands of concurrent users, and provide a user-friendly interface to ensure seamless service delivery for applicants and staff. Scalability, security, and reliability are critical to the system's success.

5.2 Software Requirements Specification (SRS) Document

1. Introduction

1.1 Purpose of this Document: To specify the software requirements for a Passport Automation System that streamlines the passport application, verification, and issuance processes, ensuring efficient service delivery and user experience.

1.2 Scope of this Document: Users can apply for passports online, track the status of their applications, and receive notifications for document submission and appointment scheduling. The system also allows staff to verify applications, manage appointments, and process passport issuance.

1.3 Overview: This document outlines the requirements and procedures involved in the automation of passport services, from application submission to passport issuance.

2. General Description

2.1 Product Perspective:

- Applicant's Perspective:
    - Submit passport applications online with personal details, upload required documents, and track the application status.

    - Schedule appointments for document verification and biometric data submission.
- Staff's Perspective:
    - Access applicant details, verify submitted documents, and schedule appointments for in-person verifications.
    - Manage passport issuance and record passport delivery status.

2.2 Product Functions:

- Submit passport applications online
- Upload required documents and track application status
- Schedule and manage appointments
- Passport verification and issuance
- Generate reports on pending and processed applications

2.3 User Characteristics:

- Users include applicants (citizens applying for passports) and staff (government personnel processing applications).
- Applicants must have basic computer literacy to submit online applications.
- Staff must be familiar with document verification and passport issuance processes.

2.4 General Constraints:

- The system must operate in compliance with the legal requirements for passport issuance.
- Users must have valid identification and provide required documentation before applying for a passport.
- The system must ensure real-time updates on application status.

2.5 Assumptions and Dependencies:

- Applicants will provide accurate personal and document information.
- The system will rely on a secure database infrastructure to store sensitive applicant data.
- Internet access will be required for online application submission and status tracking.

3. Functional Requirements

3.1 Passport Application System:

- Applicants can create new applications by submitting personal details and uploading documents.
- The system will verify the completeness of the application and schedule an appointment for document verification.

3.2 Appointment Management System:

- Applicants can schedule, view, and reschedule appointments for document submission and biometric capture.
- The system will send appointment reminders via email or SMS.

3.3 Verification and Issuance System:

- Staff can access submitted applications, verify uploaded documents, and update application statuses.
- Passport issuance can be processed once all verifications are completed.

3.4 Notification System:

- Automated notifications will be sent to applicants for appointment scheduling, status updates, and document submission reminders.

4. Interface Requirements

4.1 User Interface (UI): The system should provide a user-friendly interface for applicants to submit applications, track status, and schedule appointments.

4.2 API Interface: An API will allow integration with external systems, such as government identity verification systems and postal services for passport delivery.

4.3 Database Interface: The database interface will manage the storage and retrieval of applicant data, verification status, and passport issuance details.

5. Performance Requirements

- Response Time: The system should load and display application status updates within 3 seconds.
- Availability: The system should maintain an uptime of 99.9%, excluding scheduled maintenance.
- Error Handling: Clear and user-friendly error messages should be provided for failed application submissions or system errors.
- Security: All personal and sensitive data must be encrypted during storage and transmission to ensure privacy and data integrity.

6. Design Constraints

- Database Structure: The system should use a relational database to store structured data, such as applicant information, document submissions, and application statuses.
- Microservices Architecture: The system should be developed as microservices to allow scalability and maintainability of different functions, such as application processing and document verification.
- Integration with Existing Systems: The passport automation system must integrate with national ID verification systems, payment gateways, and postal services for passport delivery.
- Hardware Limitations: The system must be compatible with the standard server and hardware configurations of government infrastructure.

7. Non-Functional Requirements

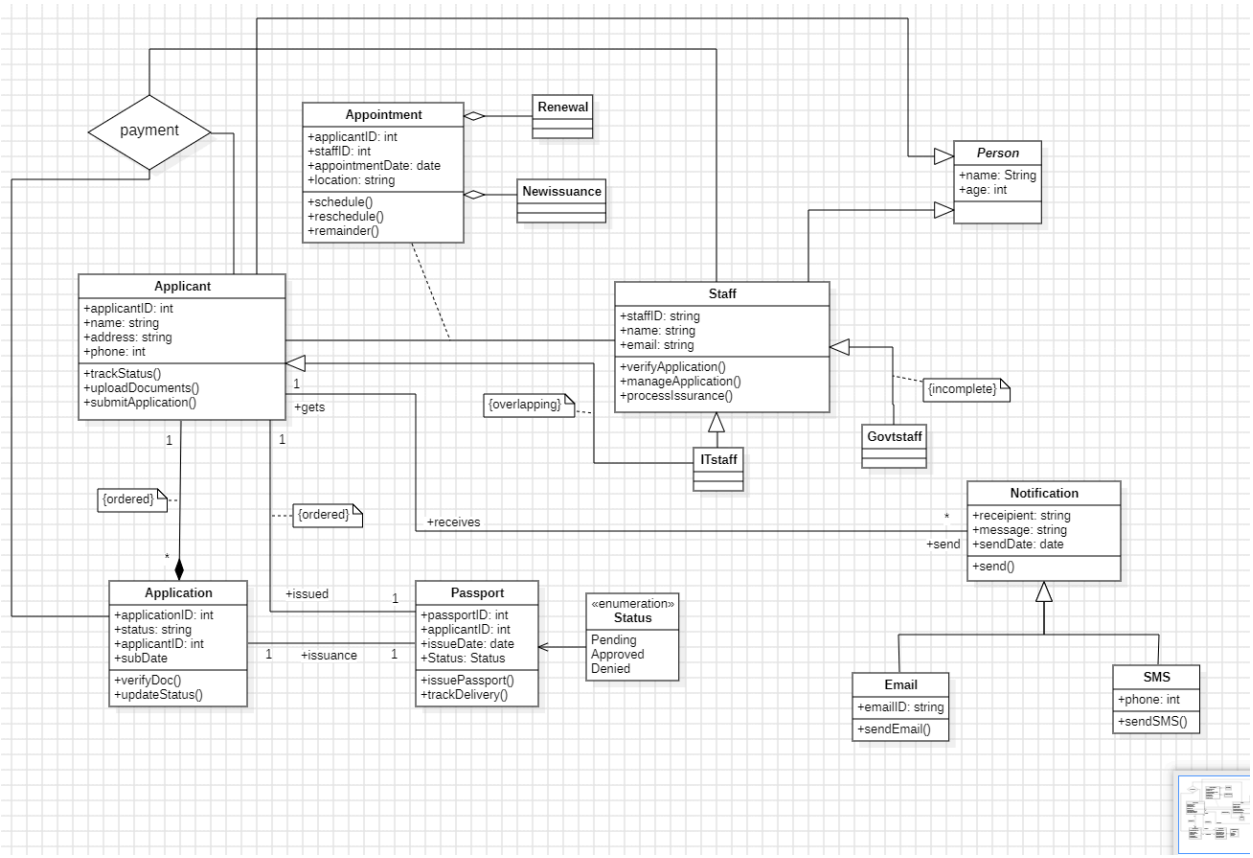
- Performance: The system should process document uploads and application submissions within 5 seconds.
- Scalability: The system must support thousands of concurrent users submitting and tracking applications.
- Security: Sensitive data such as passport details, biometric information, and personal documents must be encrypted and protected from unauthorized access.
- Usability: The system should offer an intuitive interface that requires minimal training for applicants and staff.

8. Preliminary Schedule and Budget

The passport automation system project is estimated to take approximately 6 months to complete. This timeline includes hiring staff, planning the software architecture, developing the system, testing for quality assurance, and deploying the system with user training. The total budget required is estimated to be around ₹50 lakhs, covering personnel costs (software developers, security experts, UI/UX designers, and QA testers), software and database licenses, server and infrastructure costs, training materials, and contingency for unexpected expenses.

5.3 Class Diagram

5.3.1 Advanced Class Diagram



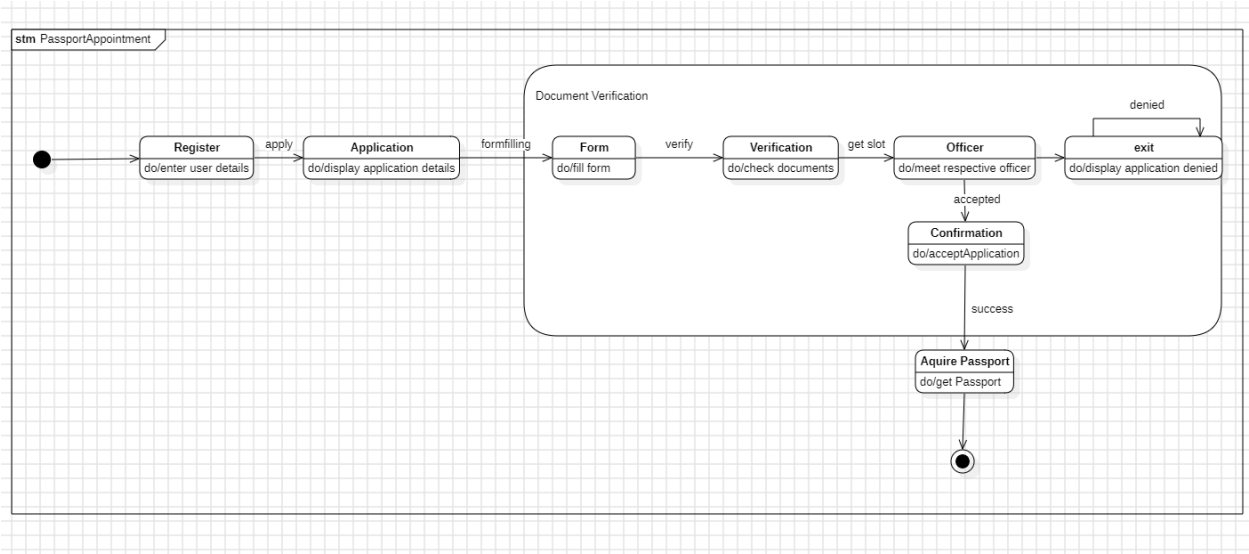
Description: This diagram models the structure of the passport automation system. Key classes include:

- Applicant: Represents users applying for a passport. Contains attributes such as ID, name, and address, along with methods to track status, upload documents, and submit applications.

- Application: Maintains details about the application, such as ID, status, and submission date. Methods include verifying documents and updating the status.
- Passport: Represents the issued passport with attributes like passport ID and issue date.
- Staff: Handles application verification and passport issuance.
- Notification: Facilitates sending updates through SMS or email.

5.4 State Diagram

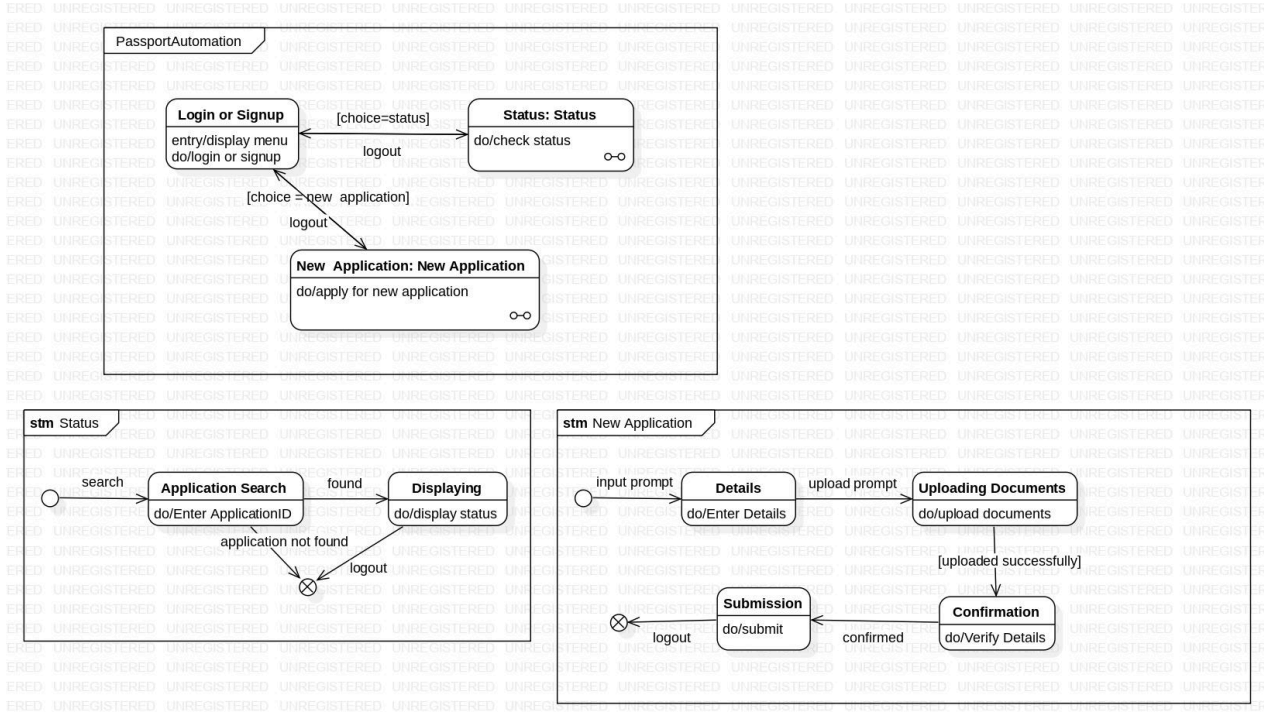
5.4.1 Simple State Diagram



Description: This diagram focuses on the appointment scheduling process:

- Register: Users input their details to begin the process.
- Application: The user views application details and proceeds to fill out forms.
- Verification: Submitted documents are verified by the system or respective officers.
- Confirmation: If verified, the application is accepted; otherwise, it exits with an error.

5.4.2 Advanced State Diagram

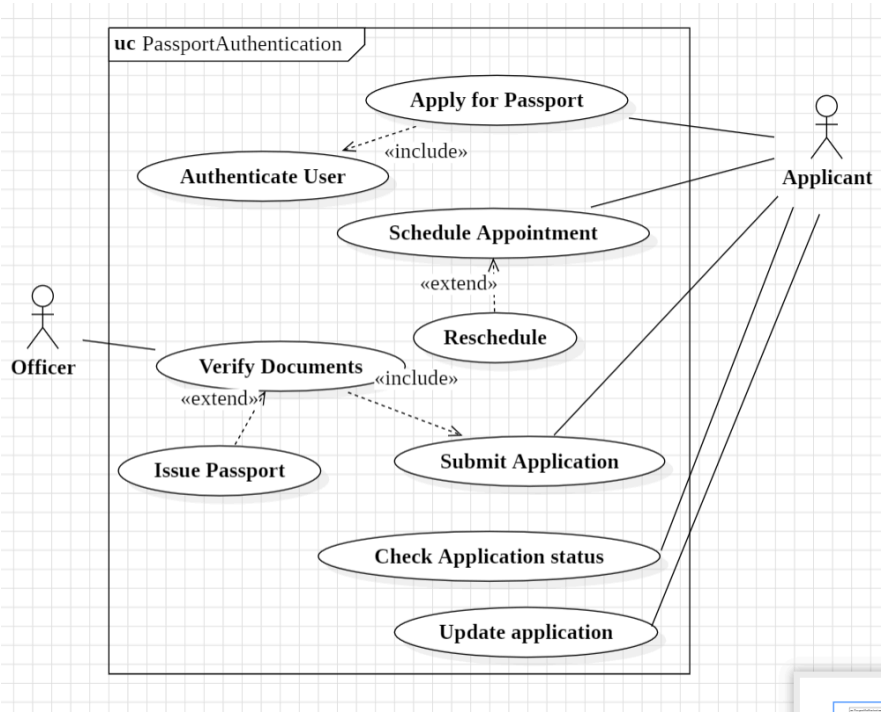


Description: This state diagram illustrates the dynamic behavior of the passport automation system. It depicts three major states:

- Login or Signup: Users can either log in or register for a new account.
- New Application: The process includes entering details, uploading required documents, submitting the application, and confirming details.
- Status Check: Users can track the application status by entering the application ID. The system confirms whether the application is found or not.

5.5 Use case diagram

5.5.1 Simple Use Case Diagram



Description:

The key actors are:

- Applicant: Responsible for applying for a passport, scheduling appointments, submitting applications, checking application status, and updating applications.
- Officer: Handles tasks like verifying documents and issuing passports.

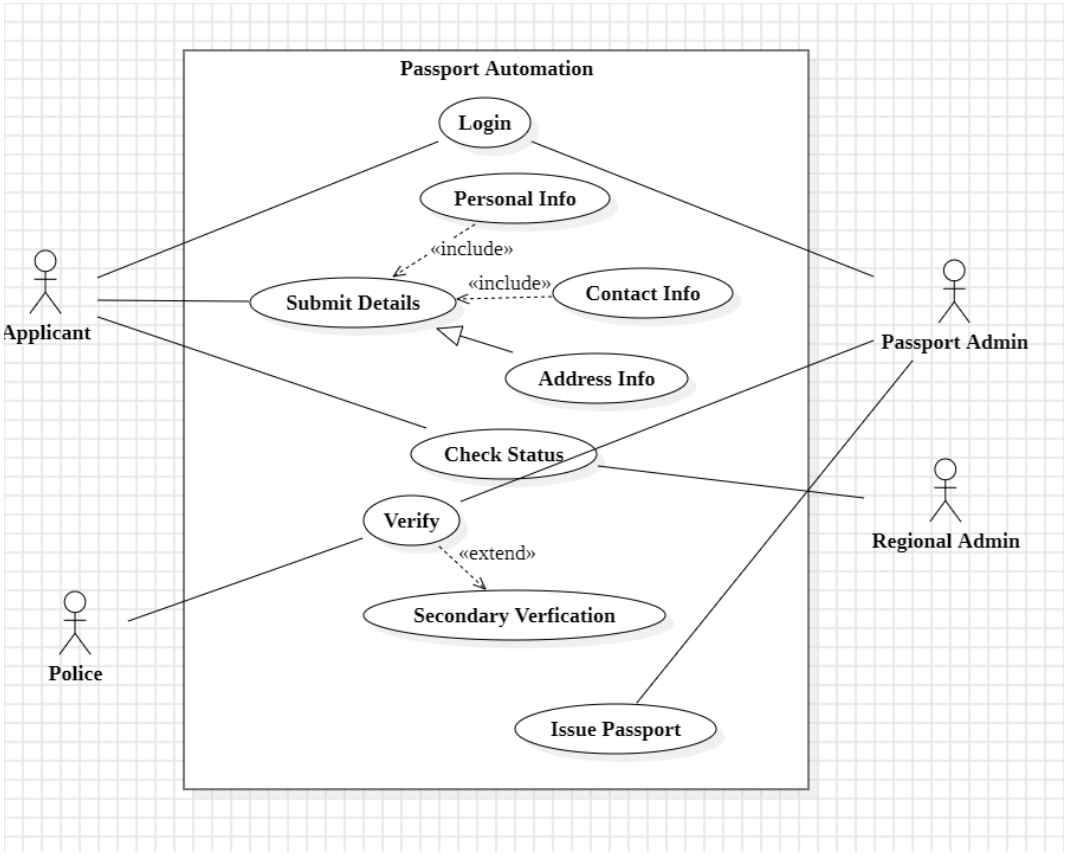
The main use cases include:

- Apply for Passport: Initiated by the applicant and includes authentication and scheduling appointments.
- Submit Application: The applicant submits their passport application for processing.
- Verify Documents: Managed by the officer, ensuring all documents are valid before proceeding.
- Issue Passport: The final step handled by the officer after successful verification.
- Check Application Status: Allows applicants to track their application progress.
- Reschedule Appointment: Extends the scheduling use case for flexibility.

Relationships:

- The diagram uses include and extend relationships to represent dependencies and optional functionalities.

5.5.2 Advanced Use Case Diagram



Description:

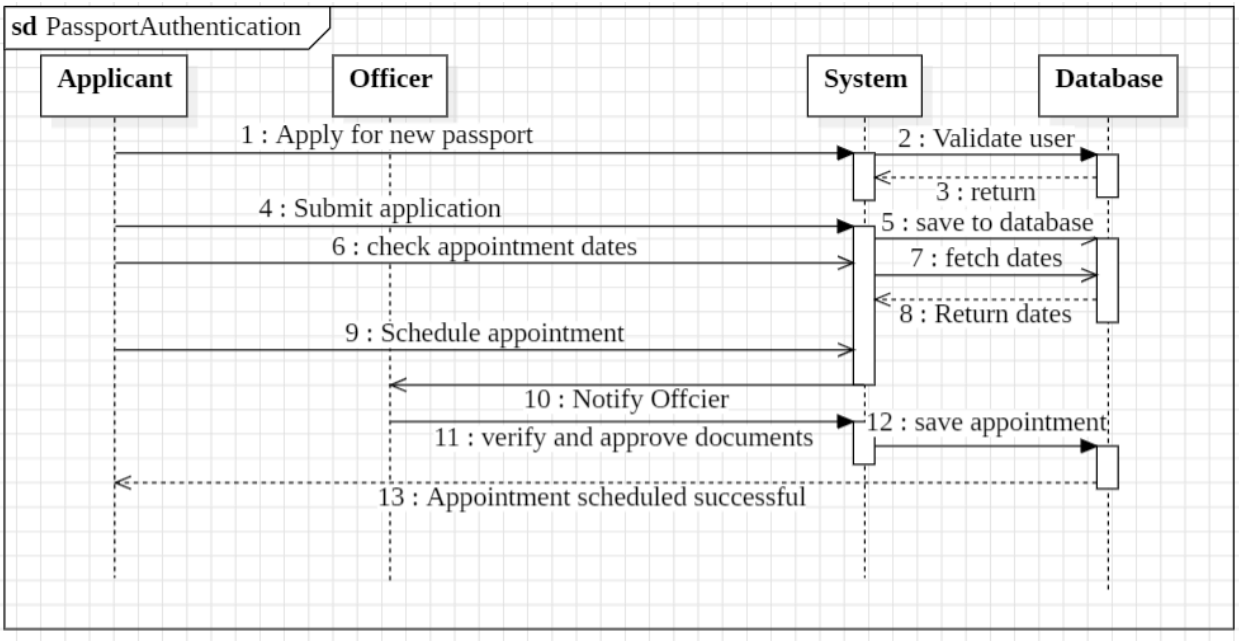
- Actors:
    - Applicant: Logs in, submits personal/contact/address information, and checks application status.
    - Passport Admin: Verifies applications.
    - Regional Admin: Handles secondary verification and passport issuance.
    - Police: Assists in verification.
- Use Cases:
    - Core functions like login, submit details, check status, verification, and passport issuance.
- Relationships: Includes (mandatory functionality) and extends (optional or conditional steps).

Additional Elements:

- Advanced verification steps and conditions for issuing passports.
- Integration of multiple user roles like Admins and Police for secure processing.

5.6 Sequence Diagram

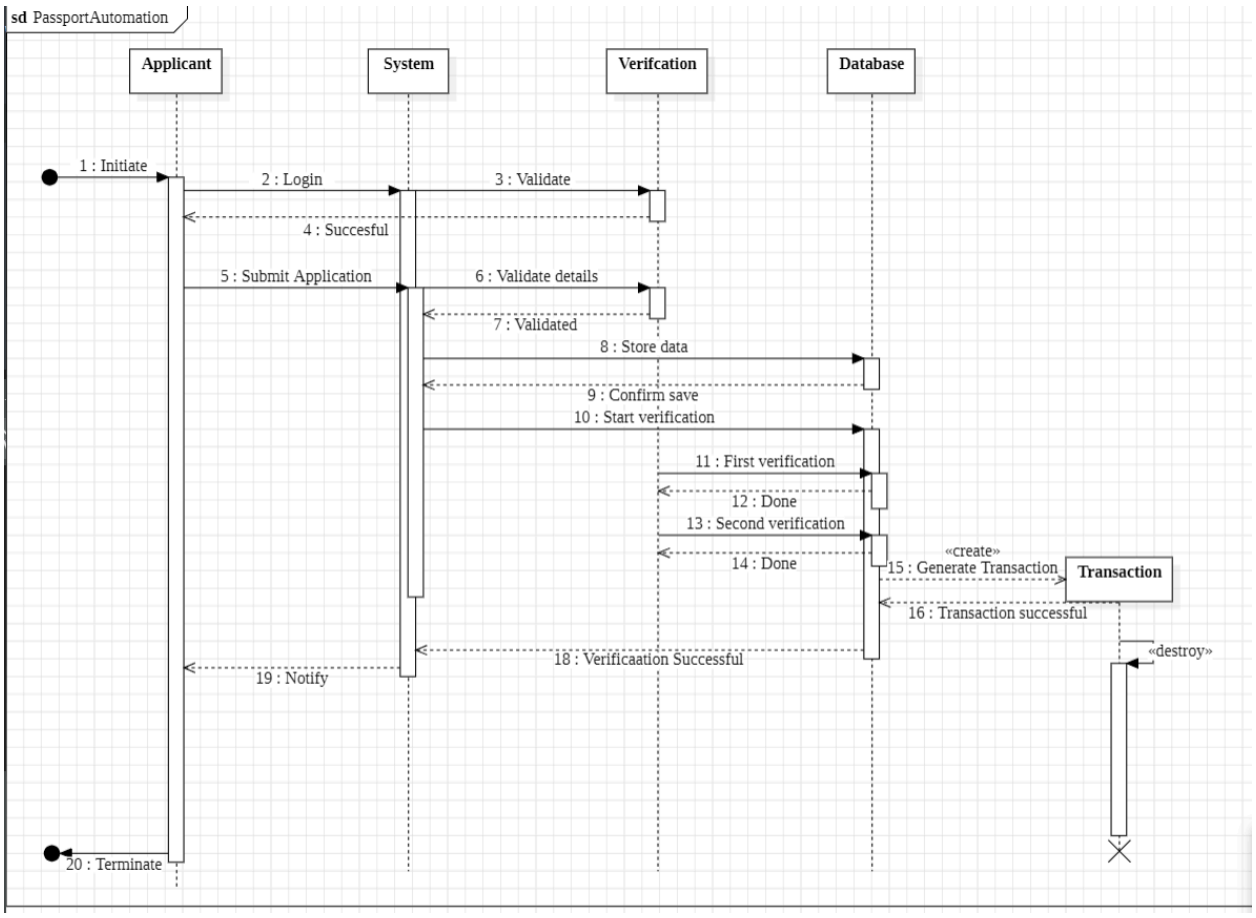
5.6.1 Simple Sequence Diagram



Description: The Sequence Diagram details the interactions between the Applicant, Officer, System, and Database during the passport application process. Key steps include:

1. The applicant applies for a passport, which triggers user validation by the system.
2. The validated user submits the application, and the data is saved in the database.
3. The applicant checks appointment dates, which are fetched and returned by the system.
4. An appointment is scheduled and saved, with the officer notified for further processing.
5. The officer verifies and approves documents, leading to a successful appointment confirmation.

5.6.2 Advanced Sequence Diagram



Description:

Key Elements:

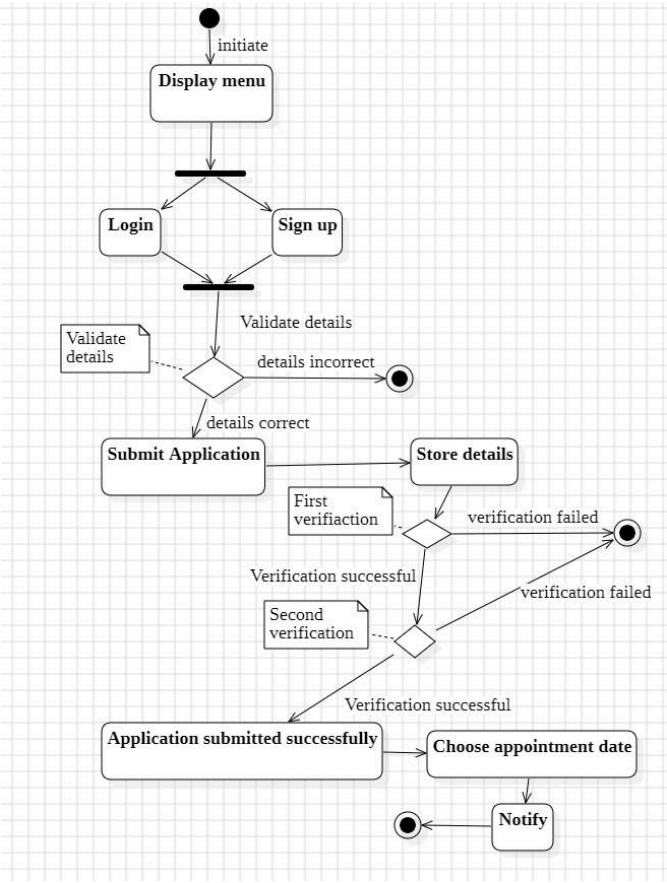
- Actors: Applicant, System, Verification module, Database, and Transaction service.
- Flow:
    1. Applicant logs in.
    2. Application is submitted and validated by the system.
    3. Details are stored in the database.
    4. Verification proceeds in two stages.
    5. Transaction is generated upon successful verification.
    6. Applicant is notified of the process status.

Additional Elements:

- Loops for repeated verification attempts.
- Conditional paths for data correction and system failure scenarios.

5.7 Activity Diagram

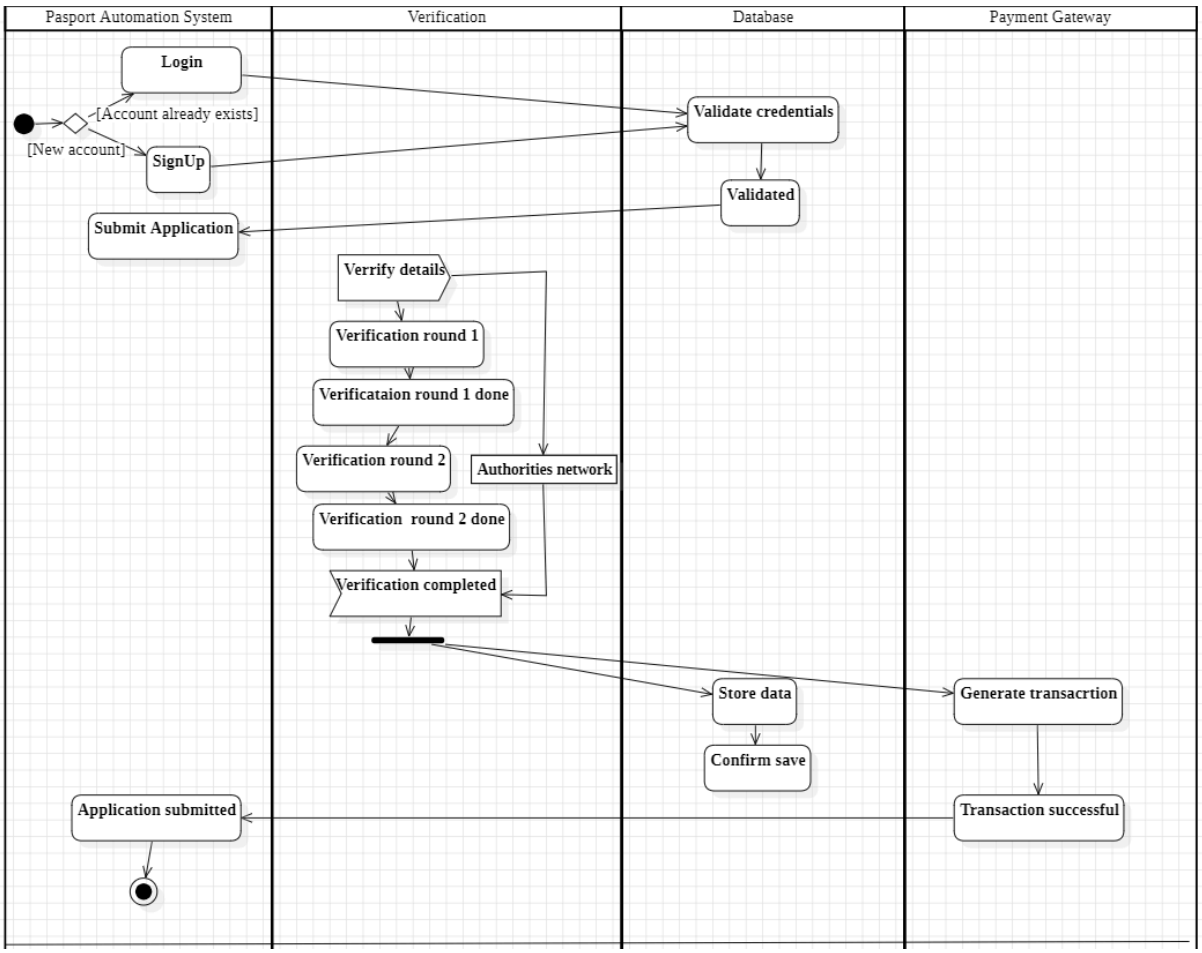
5.7.1 Simple Activity Diagram



Description: The Activity Diagram represents the flow of activities in the Passport Authentication System, starting from the display menu:

1. Login/Sign Up: Users must either log in or sign up. Validation ensures correct user details.
2. Submit Application: Once validated, users proceed to submit their application.
3. Store Details: Application details are stored, followed by two levels of verification.
4. Verification: If both verification levels succeed, the application is marked as successfully submitted.
5. Appointment Scheduling: The user selects an appointment date and is notified accordingly

5.7.2 Advanced Activity Diagram



Description:

Key Elements:

- Swimlanes: Divides actions across:
    - Passport Automation System: Login, sign-up, and application submission.
    - Verification: Two rounds of data verification.
    - Database: Credential validation and data storage.
    - Payment Gateway: Transaction generation.
- Flow:
    - If the account exists, login proceeds; otherwise, new accounts are created.
    - Submitted details undergo verification rounds before being saved.
    - Transaction completes the process.